PARALLEL SUBSTEPPING SCHEME FOR ELASTO-PLASTIC FINITE ELEMENT ANALYSIS

Zhongwen Ding

B.E. (Northeastern University, P. R. China)

September 1999

A thesis submitted for the degree of Master of Engineering of The Australian National University

Department of Engineering
Faculty of Engineering and Information Technology
The Australian National University

To my parents

Declaration

This thesis contains no material which has been previously accepted for the award of any other degree or diploma in any university, institute or college, and contains no material previously published or written by another person, except where due reference is made.

Canberra, September 1999.

Zhongwen Ding
Department of Engineering
Faculty of Engineering and Information Technology
The Australian National University
Canberra ACT 0200, AUSTRALIA.

Journal Papers:

[J1]Z. Ding, Dr. S. Kalyanasundaram, Dr. M. Cardew-Hall, Dr. S. Robert and Dr. L. Grosz. "*Integration Algorithms for Three-dimensional Elasto-plastic Finite Element Analysis*," Submitted to *International Journal of Numerical Methods in Engineering*, under review.

[J2]Z. Ding, Dr. S. Kalyanasundaram, Dr. L. Grosz, Dr. M. Cardew-Hall and Dr. S. Robert. "*Efficient Parallel Algorithms for Elasto-Plastic Finite Element Analysis*," Submitted to the journal of *Computers and Structures*, under review.

Conference Papers:

[C1]Z. Ding, Dr. S. Kalyanasundaram, Dr. L. Grosz, Dr. M. Cardew-Hall and Dr. S. Robert, "*Parallel Substepping Schemes for Elasto-Plastic Finite Element Stress Analysis*," The Second Australasian Congress on Applied Mechanics (ACAM 99), Feb. 10 - 12, 1999, Canberra, Australia

[C2]Z. Ding, Dr. S. Kalyanasundaram, Dr. L. Grosz, Dr. M. Cardew-Hall and Dr. S. Robert, "*Application of Parallel Conjugate Gradient Method to Non-linear Problem in Solid Mechanics*," International Conference on Applied Modelling and Simulation (AMS'99), Sep. 1-3, 1999, Cairns, Queensland, Australia

[C3]Z. Ding, Dr. S. Kalyanasundaram, Dr. L. Grosz, Dr. M. Cardew-Hall and Dr. S. Robert, "*Development of A New Method for Solving the Initial Value Problem in Elasto-Plastic Deformation Analysis*," Submitted to the 9th Biennial Computational Techniques and Applications Conference and Workshops (CTAC99), September 20–24, 1999, Canberra, Australia

[C4]Z. Ding, Dr. S. Kalyanasundaram, Dr. L. Grosz, Dr. M. Cardew-Hall and Dr. S. Robert, "*Parallel Elasto-Plastic Finite Element Analysis in a Workstation Cluster Environment*," Submitted to The 7th International Symposium on Structural Failure and Plasticity (IMPLAST 2000), Oct. 4-6, 2000, Melbourne, Australia

Abstract

In this thesis, we present the development of the parallel algorithms for elasto-plastic analysis by using finite element method. The method is based on dividing the original structure into a number of substructures which are treated as isolated finite element models via the interface conditions. Separate input and output files are established for each subdomain. These files are read from and written to by local copies of the program executable operating in parallel. After reading corresponding input file, each processor generates the substructure in parallel on which it will operate without any need for communication. During the overall solution, each processor performs identical instructions, but on different sets of data.

We focus on the establishment of algorithms for integration of the strain and stress relations and solution of the resulting systems of equations. We employ a parallel substructure oriented preconditioned conjugate gradient method combined with minimal residual smoothing and the diagonal storage scheme to solve the systems of equations. The solution method proposed does not require the formation of global system of equations, but computes directly the displacements for each substructure, as opposed to solving a global system of nodal equations. Throughout the analysis, each processor stores only the information relevant to its substructure and generates the local stiffness matrix.

After the displacements are calculated a substepping scheme is used to integrate elasto-plastic stress-strain relations. The procedure outlined controls the error of the computed stress by choosing each substep size automatically according to a prescribed tolerance. The results indicate that the combination of this substepping scheme and the stress correction which is applied at the end of integration process can increase both accuracy and efficiency significantly.

When we implement the parallel algorithms we have to address the problems of load balancing and interprocessor communication, etc. The method we use to balance the load is to employ different partitioning schemes. The interprocessor communication is optimized by using a special element numbering and an optimal communication scheme.

In this thesis we will describe the implementation in further detail and give some examples of results obtained from experimental runs via Message Passing Interface on the Linux-Alpha workstation cluster at the Australian National University Supercomputer Facility. The results are designed to highlight the performance of the algorithms developed as well as their efficiency on a parallel machine.

Acknowledgements

Working on this project has been a source of great pleasure for me. At this juncture, I would like to acknowledge the people who worked with me and helped make this project a reality.

I would like to thank the people in the Department of Engineering at the Australian National University. They have helped to create a good working environment, both academically and socially. I would especially like to thank my supervisor, Dr. Shankar Kalyanasundaram for his patience and guidance. I am most indebted to him for his support throughout the duration of this project.

I would like to thank the staff at the Australian National University Supercomputer Facility, especially Roger Brown and David Singleton for their technical advice on parallel computing, particularly with regards to the Alpha-Linux workstation cluster.

I would also like to thank Dr. Lutz Grosz and Dr. Stephen Roberts, at the School of Mathematical Science, and Dr. Mick Cardew-Hall, head of the Department of Engineering, for their valuable input into this project.

Most of all, I would like to thank my wife, Xiaoli, for her affectionate support, patience, and encouragement all through my education at ANU.

Contents

List of Figures

List of Tables

Notation

F	yield function
F^e	equivalent nodal forces
E	Young's modulus
H	linear hardening parameter
H'	derivative of the hardening function
Q	plastic potential
N	shape functions
D	elastic matrix
D_{ep}	elasto-plastic matrix
B	elastic strain matrix
J_i	stress invariants
J_i'	invariants of the deviatoric stresses
K^e, K	stiffness matrix (element/global)
C	preconditioning matrix
T	time
T_s	serial run time
T_p	parallel run time
ΔT	substep size
A	transformation matrix
W	potential energy of loads
W_p	total plastic work
R	residual in domain
S	speedup
E	efficiency

a	flow vector
r	residual
p	direction vector
c	cohesion parameter
x, x^e	displacement vector (global/element)
δd	virtual displacement
δu	internal displacement
f	applied forces
b	distributed loads/unit volume
ε	strain vector
$\Delta\varepsilon$	strain increment
$\Delta\varepsilon_e$	elastic strain increment
$\Delta\varepsilon_p$	plastic strain increment
$\delta\varepsilon$	virtual strain
$\bar{\varepsilon}_p$	effective plastic strain
λ	plastic multiplier
ν	Poisson's ratio
κ	hardening parameter
σ	stress vector
$\Delta\sigma$	stress increment
σ_e	trial stress
$\sigma_Y{}^o$	uniaxial yield stress
$\bar{\sigma}$	effective stress
$\hat{\sigma}$	stress, finite element approximation
$\Delta\tau$	shear stress increment
ϕ	angle of internal friction
Ψ	residual force vector

Chapter 1

Introduction

1.1 The Research Problem

The finite element method is now firmly accepted as a powerful general technique for the numerical solution of a variety of problems encountered in engineering. Among its many applications, the finite element analysis of elasto-plastic behaviour is a subject of great importance for fundamental and practical reasons. Elasto-plastic modelling can help us make a more complete use of the strength resources of solids and leads to an efficient method for calculating details of machines and structures as regards to their load-bearing capacity. However, as increasingly large-scale three-dimensional finite element models are currently being used in various disciplines for realistic simulation of engineering problems, the use of such complex models raises a number of questions in relation to accuracy and efficiency. A great deal of effort has therefore been invested in developing algorithms that can quickly and accurately solve the large-scale problems encountered in practice.

The analysis of nonlinear elasto-plastic problems must proceed in an incremental manner since the solution at any stage may not only depend on the current displacement of the structure, but also on the previous loading history. A simplified depiction of sequential program structure is given in Figure 1-1. The diagrams on the right-hand side of Figure 1-1 indicates the computing time spent on different parts of the program. For large, three-dimensional problems the overall CPU time is dominated by the solution of systems of equations. The only other parts of the program that require significant computational resources are the computation of the stiffness matrices for each

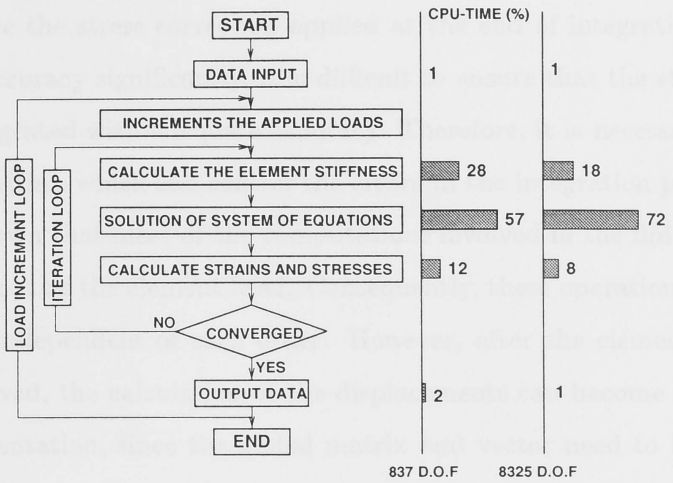

Figure 1-1: Program structure for elasto-plastic analysis

element and evaluation of strains and stresses for each integration point. For large-scale three-dimensional problem, the integration of strain-stress relations and the solution of equations form two important stages in finite element elasto-plastic analysis.

As shown in Figure 1-1, two primary loops are necessary to increment the applied loading and to iterate the solution until convergence occurs. If the load increment is too small, the total number of iterations involved in the overall solution will increase dramatically. The computational time will increase as well since during each iteration a linear system of equations has to be solved which is the most time-consuming part in the overall solution of large-scale three-dimensional applications. On the other hand, in each load increment, the inaccurate integration of constitutive equations may cause the prolonged iteration to converge. This indicates that efficient algorithms for elasto-plastic finite element analysis are essential to allow greater load increment and to decrease the total number of iterations needed to converge. Therefore, one of the key factors of an efficient algorithm is that the constitutive law be integrated accurately.

The conventional method for integrating elasto-plastic stress-strain relations used simple Euler scheme and divided the integration process into a number of equal sub-steps. However, this technique requires that load increments be kept small so that the stresses computed at the end of integration procedure do not deviate too far from the yield surface. For the applications with relative large load increments, the computed stresses may not satisfy the yield criterion after integration process. Therefore, a correction of stresses which is applied at the end of integration process has been used

frequently. Since the stress correction applied at the end of integration process does not affect the accuracy significantly, it is difficult to ensure that the strain-stress relation can be integrated with adequate accuracy. Therefore, it is necessary to develop a integration algorithm which can control the errors in the integration process.

It is well-known that most of the computations involved in the finite element analysis are carried out at the element level. Consequently, these operations can be carried out in parallel independent of each other. However, after the elemental calculations have been achieved, the calculation of the displacements can become a bottleneck for parallel implementation, since the global matrix and vector need to be assembled at this stage. It would be very natural, especially from a parallel processing viewpoint, if the formation of the global equations could be avoided, that is, the selected numerical algorithm could be operated at the element or substructure level. Generally, this is the case for iterative solution methods based on matrix-vector. Also, since the iterative method employed in the finite element analysis does not change the structure of stiffness matrix, it maintains its sparsity. Hence, the computational costs and memory space associated with zero fill-ins can be greatly reduced by using a suitable storage scheme. So development of a efficient parallel algorithm is another important task for this research work.

1.2 Literature Review

1.2.1 Integration Algorithm for Elasto-Plastic Problems

In the past 20 years, significant advances have been made in the development and application of numerical methods to the solution of elasto-plastic problems. It is well known that one of the simplest numerical schemes, which has been used widely in finite element codes, is the first order Euler algorithm. This has the advantage of being straightforward but also has the disadvantage of being accurate only for very small time steps. To avoid this shortcoming, it is usual to subdivide the particular time step into a number of smaller substeps and apply the Euler schemes to each of these[1, 2]. This partially overcomes the disadvantage of the Euler approach but usually leads to a set of stresses which do not lie precisely on the yield surface at the end of each time step. Since these errors are cumulative, and may lead to unacceptable results in subsequent

computation, it is usual to apply some form of correction[3] to the stresses to restore them to the yield surface. Since the stress correction is normally applied at the end of integration process, it does not significantly affect the accuracy.[4]

A number of new integration algorithms therefore were developed with an aim to control the error in the integration process[4]-[26]. Wissmann and Hauck[4] developed a algorithm with an aim to control the errors in the integration process by using Richardson extrapolation to selected number of fixed size substeps. Polat and Dokainish[5] took into account the change in the plastic flow direction due to continuing plastic deformation and an automatic subincrementation scheme has been proposed for further accuracy improvement. Eterovic and Bathe[6] presented a hyperelastic-based large strain formulation using the product decomposition of the deformation gradient into elastic and plastic parts for metal plasticity. Pezeshk and Camp[7] have proposed an integration method based on a Modified Trapezoidal rule Method. The resulting algorithm is extremely simple to use. However, it is conditionally stable. It is especially worthy noting that Sloan[8, 9] used a substepping scheme to integrate the stress-strain relations. The substep size is decided by comparing a prescribed tolerance with an estimate of the error of the integrated stress increment. This error estimate is obtained by comparing the estimated stress increments which result from two integration procedures with truncation errors of different order. Two substepping schemes are recommended in [8], which are based on modified Euler method and the fifth order Runge-Kutta-England, respectively. The application of a smooth rigid strip footing resting on an ealsto-plastic soil mass indicated that no form of the stress correction is required. However, Gens and Potts suggested in [10] that, the deviation from the yield surface is directly related to the level of error in computed stresses and it is practically independent of the integration scheme adopted. When the model involves the hardening, the drift is found to be generally more pronounced. Since such discrepancies are usually cumulative, it is important to ensure that the stresses are corrected back to the yield surface at any time. In [10], it was found that even if only a single step is used with stress correction accurate results are obtained. Based on above ideas, it can be proposed that the substepping scheme with larger tolerance combined with stress correction applied at the end of integration process could be the best integration algorithm for elasto-plastic problems.

1.2.2 Parallel Elasto-Plastic Finite Element Analysis

At present, the parallel algorithms for linear analyses have made considerable headway. However, in the aspect of nonlinear analyses, wide-ranging research has not been undertaken; Wilson and Farhat[33], Sun and Mao[34], Farhat and Crivell[35], Shivakumar, Bigelow and Newman[36], Kacou and Parsons[37], Hu[38], Klaas, Kreienmeyer and Stein[39], Feriani, Franchi and Genna[40][41] *et al.* have implemented such nonlinear analysis on parallel computer systems. However the integration schemes they employed are based on the conventional finite element method in which no measure is used to control the error in the integration process. In fact, the error in the computed stresses is very important for the non-linear elasto-plastic analysis, since large error can cause a prolonged iteration to converge.

In parallel finite element analysis for elasto-plastic problem, solving the linearized system of equations forms another important stage. Techniques to solve the equations system may generally be classified into two categories, one is the direct parallel solution; Melosh and Utku[42], Doi and Koyama[43], Noor, Kamel and Fulton[44], Farhat and Wilson[45], Malone[46], Goehlich, Komzsik and Fulton[47] *et al.* have all conducted some research work in this field. The other is the iterative solution; Hughes, Levit and Winget[48], Law[49], King and Sonnad[50], Carter, Sham and Law[51], Johnsson and Mathur[52], Kumar, Grama, Gupta and Karypis[53] *et al.* have proposed different iterative algorithms. The main virtue of direct parallel solution is the strong numerical stability, but the weak point is that synchronous control must be introduced in most cases. The main advantage of iterative parallel solution is that the excessive synchronous control steps can be avoided, but there are some problems in the numerical stability of the algorithm. The direct parallel algorithms are commonly developed from the substructuring techniques or the synchronous control solutions of the finite element equations. The iterative parallel algorithms are generally based on preconditioned conjugate gradient PCG) or Jacobi methods.

1.3 Aims

The aim of this research project is to research methods for developing an effective high performance computational strategy for performing on entire

finite element solution procedure with relevance to elasto-plastic modelling.

In order to develop the above framework the following important areas needed to be researched:

- Accurate integration algorithm for strain-stress relationships.
- Efficient parallel algorithm for the solution of linear system of equations.
- A suitable storage scheme for overall solution.
- Implementation of the developed parallel algorithms on the computer platforms to achieve optimal performance.

With these thoughts in mind, we will develop a substepping scheme for elasto-plastic stress integration process. The resulting algorithm can be applicable to a general type of constitutive law and controls the error in the integration process by adjusting the size of each substep automatically in accordance with the behaviour of the constitutive law. For the equation solution, a combination of preconditioned conjugate gradient method with minimal residual smoothing will be employed. In the resulting parallel algorithm, the formation of the global system matrix is not performed, but the displacements for each substructure are computed directly, as opposed to solving a global system of nodal equations. The resulting algorithms will be tested on a workstation cluster. To obtain an optimal performance, a diagonal storage scheme will be employed. Different partitioning scheme will be used with an aim to obtain better load balance. Also, a special numbering of the elements and an optimized communication scheme are adopted in this research work.

1.4 Organisation of the thesis

This thesis chronicles our experience with parallel finite element analysis in the context of non-linear elasto-plastic finite element simulation over the past two years. An outline of this thesis is as follows.

Chapter 2 commences with a brief review of finite element method and its application to nonlinear problem. For elasto-plastic applications to be considered, basic theoretical formulations are developed in a form suitable for numerical solution. Conventional method for integrating strain-stress relations will be introduced. Following it,

an advanced substepping scheme will be introduced and its advantages will be demonstrated. The overall solution method will also be presented for completeness.

Chapter 3 deals with the development of parallel algorithm for the solution of linear system of equations. Both sequential algorithm and parallel algorithm are outlined. A finite element transformation relations is introduced which forms a bridge between sequential algorithm and parallel algorithm.

In Chapter 4, we present an introduction of parallel environment which is used in this research work. Some of the important issues in implementation of the parallel algorithms on the chosen parallel machine are also reviewed.

Chapter 5 conducts application of the resulting parallel algorithms to a three-dimensional elasto-plastic analysis. The speedup, efficiency and scalability will be studied.

Chapter 6 summarises the main conclusions of the study and provides some recommendations for the further research in this area.

Chapter 2

Finite Element Elasto-Plastic Analysis

A general introduction of finite element method is presented in this chapter. Basic theoretical formulations for elasto-plastic problems are developed in a form suitable for numerical solution. Based on it, a conventional method for integrating the strain-stress relations will be discussed. An advanced substepping scheme will be developed and its performance will be given. For a better understanding of the performance of substepping scheme, the overall solution methods which are employed in the finite element code will be briefly introduced.

2.1 Finite Element Method

The finite element method is a numerical analysis technique for obtaining approximate solution to a wide variety of engineering problems. Although originally developed to study the stresses in complex airframe structures, it has since been extended and applied to the broad field of continuum mechanics. Because of its diversity and flexibility as an analysis tool, it is receiving much attention in engineering schools and in industry.

2.1.1 History of Finite Element Method

Finite element analysis was first developed in 1943. During its early development for stress analysis problems the method relied heavily on a physical interpretation in which the structure was assumed to be composed of elements physically connected only at

a number of discrete nodal points. Later the application of the method to structural mechanics problems was developed through the use of the principle of virtual work and energy methods. The method was then generalised and its wider mathematical roots were recognised. It was shown that finite elements could be applied to any mathematical problem for which a variational functional existed. More recently, finite element solutions have been developed which are based on the well known, classical techniques known as "weighted residual methods". Since the 1970's the rapid growth in engineering usage of computer technology has a significant effect upon the acceptance of the finite element method. In fact the finite element method is now firmly established as an engineering tool of wide applicability. One of the principal advantage of the finite element method is the unifying approach it offers to the solution of diverse engineering problems.

2.1.2 General Procedures of Finite Element Method

Regardless of the approach used to different applications, the solution of finite element method always follows an orderly step-by-step process. To summarize in general terms how the finite element method works we will succinctly list these steps. These well-defined modules are a base of parallel finite element analysis.

1. ***Discretize the continuum.*** The first step is to divide the continuum or solution region into elements. A variety of element shapes is available for different analysis.
2. ***Select interpolation functions.*** The next step is to assign nodes to each element and then choose the type of interpolation function to represent the variation of the field variable over the element.
3. ***Find the element properties.*** Once the finite element model has been established, we are ready to determine the matrix equations expressing the properties of the individual element.
4. ***Assemble the element properties to obtain the system equations.*** To find the properties of the overall system modelled by the network of elements we must "assemble" all the element properties.
5. ***Solve the system equations.*** The assembly process of the preceding step gives a set of simultaneous equations that we can solve to obtain the unknown nodal

values of the field variable. Technique to solve this equations system may generally be classified into direct and iterative methods.

6. ***Calculate the element characteristics.*** Normally we use the solution of the system equations to calculate other important parameters, such as strains, stresses,...etc.

2.1.3 Basic Formulation of Finite Element Method for Non-Linear Problem

For any numerical approach an approximate solution is attempted by assuming that the behaviour of the continuum can be represented by a finite number of unknowns. As previously mentioned in the finite element method the continuum is divided into a series of elements which are connected at a finite number of points known as nodal points.

For structural applications at least, the governing equilibrium equations can be obtained by the principle of virtual work[2]. Consider the solid, in which the internal stresses σ, the distributed loads/unit volume b and external applied forces f form an equilibrating field, to undergo an arbitrary virtual displacements pattern δd which results in compatible strains $\delta\varepsilon$ and internal displacements δu. Then the principle of virtual work can be expressed as

$$\int_{\Omega}(\delta\varepsilon^T\sigma - \delta u^T b)d\Omega - \delta d^T f = 0 \tag{2.1}$$

In the finite element displacement method, the displacement is assumed to have unknown values only at the nodal points, so that the variation within any element is described in terms of the nodal values by means of interpolation functions. Thus

$$\delta u = N\delta d \tag{2.2}$$

where N is the set of interpolation functions termed the *shape functions*. The strains within the element can be expressed in terms of the element nodal displacements as

$$\delta\varepsilon = B\delta d \tag{2.3}$$

where B is the strain matrix generally composed of derivatives of the shape functions. Then the element assembly process gives

$$\int_{\Omega}\delta d^T(B^T\sigma - N^T b)d\Omega - \delta d^T f = 0 \tag{2.4}$$

where the volume integration over the solid is the sum of the individual element contributions. Since this expression must be hold true for any arbitrary δd value

$$\int_{\Omega} B^T \sigma d\Omega - f - \int_{\Omega} N^T b d\Omega = 0 \tag{2.5}$$

For the solution of nonlinear problems which will be described in the following, Equation 2.5 will not generally be satisfied at any stage of the computation, and

$$\Psi = \int_{\Omega} B^T \sigma d\Omega - \left(f + \int_{\Omega} N^T b d\Omega \right) \neq 0 \tag{2.6}$$

where Ψ is the residual force vector. For an elasto-plastic situation the material stiffness is continually varying, and instantaneously the incremental stress/strain relationship is given

$$\Delta\sigma = D_{ep} \Delta\varepsilon \tag{2.7}$$

For the purpose of evaluating the material tangential stiffness matrix K_T at any stage, the incremental form of (2.6) must be employed. Thus, within an increment of load we have

$$\Delta\Psi = \int_{\Omega} B^T \Delta\sigma d\Omega - \left(\Delta f + \int_{\Omega} N^T \Delta b d\Omega \right) \tag{2.8}$$

Substituting for $\Delta\sigma$ from Equation 2.7 results in

$$\Delta\Psi = K_T \Delta d - \left(\Delta f + \int_{\Omega} N^T \Delta b d\Omega \right) \tag{2.9}$$

where

$$[K_T] = \int_{\Omega} B^T D_{ep} B d\Omega \tag{2.10}$$

is termed the *element stiffness matrix*. The final system of equations that results from the above approximation is of the form

$$[K]\{x\} = \{f\} \tag{2.11}$$

where the global stiffness matrix $[K]$ is really a collection of elemental stiffness matrices

$$[K] = \sum_{e=1}^{N} [K^{(e)}] \tag{2.12}$$

These equations are then solved by any standard technique to yield the nodal displacements. After this, the stresses within each element can then be calculated from the nodal displacements.

2.1.4 Applications of Finite Element Method

The range of possible applications of the finite element method extends to all engineering disciplines, although civil and aerospace engineers concerned with stress analysis are the most frequent user of the method. Its applications mainly range from the stress analysis of solids to the solution of acoustical, neutron physics and fluid dynamics problems. Indeed the finite element process is now established as a general numerical method for the solution of partial differential equation systems, subject to known boundary and/or initial conditions.

For linear application, at least, the technique is widely employed as a design tool. Similar acceptance for nonlinear applications is dependent on two major factors. Firstly, in view of the increased numerical operations associated with nonlinear problems, considerable computing power is required. Although developments of high-speed digital computers in the last decade or so met this need to some extent, in fact, some problems can still be classified as 'difficult' to solve, even for modern vector supercomputers. Parallel processing may be the most promising way to reduce the computation time as new generations of parallel computers are rapidly emerging. Secondly, before the finite element method can be used in design, the accuracy of any proposed solution technique must be proven. Although the development of improved element characteristics and more efficient nonlinear solution algorithms and the experience gained in their application to engineering problems have ensured that nonlinear finite element analyses can now be performed with some confidence, efficient and accurate algorithms for parallel processing are still urgently needed. These suitable algorithms are essential to allow greater load step and to decrease the total amount of computation. Hence the effort on trying to remove the barriers to the common use of nonlinear finite element techniques will never end.

2.2 The Mathematical Formulation of Elasto-Plastic Problem

In this section, we consider the elasto-plastic stress analysis of solid which conforms to three-dimensional conditions. The basic laws governing elasto-plastic material behaviour in a three-dimensional solid must be presented before the numerical aspects of

the problem can be considered and to this end some concepts, such as the plastic potential and the flow rule will be introduced. Only essential expressions will be provided in this thesis and a more complete theoretical treatment can be found, for example, in [2] and [29].

The object of the mathematical theory of plasticity is to provide a theoretical description of the relationship between stress and strain for a material which exhibits an elasto-plastic response. In essence, plastic behaviour is characterised by an irreversible straining which is not time dependent and which can only be sustained once a certain level of stress has been reached. In this thesis we outline the basic assumptions and associated theoretical expressions for a general continuum. In order to formulate a theory which models elasto-plastic material deformation three requirements have to be met:

- An explicit relationship between stress and strain must be formulated to describe material behaviour under elastic conditions,i.e. before the onset of plastic deformation.
- A yield criterion indicating the stress level at which plastic flow commences must be postulated.
- A relationship between stress and strain must be developed for post-yield behaviour,i.e. when the deformation is made up of both elastic and plastic components.

Before the onset of plastic yielding the relationship between stress and strain is given by the standard linear elastic expression.

$$\Delta\varepsilon = D^{-1}\Delta\sigma \tag{2.13}$$

For three-dimensional isotropic problems

$$\{\Delta\sigma\} = (\Delta\sigma_x, \Delta\sigma_y, \Delta\sigma_z, \Delta\tau_{xy}, \Delta\tau_{yz}, \Delta\tau_{zx})^T \tag{2.14}$$

$$\{\Delta\varepsilon\} = (\Delta\varepsilon_x, \Delta\varepsilon_y, \Delta\varepsilon_z, \Delta\gamma_{xy}, \Delta\gamma_{yz}, \Delta\gamma_{zx})^T \tag{2.15}$$

$$D = \begin{bmatrix} A_1 & A_2 & A_2 & 0 & 0 & 0 \\ & A_1 & A_2 & 0 & 0 & 0 \\ & & A_1 & 0 & 0 & 0 \\ & & & A_3 & 0 & 0 \\ & & & & A_3 & 0 \\ sym. & & & & & A_3 \end{bmatrix} \tag{2.16}$$

where

$$A_1 = \frac{E(1-\nu)}{(1+\nu)(1-2\nu)}; \quad A_2 = \frac{E\nu}{(1+\nu)(1-2\nu)}; \quad A_3 = \frac{E}{2(1+\nu)}$$

in which E and ν are respectively the elastic modulus and Poisson's ratio of the materials.

2.2.1 The Yield Criterion

The yield criterion determines the stress level at which plastic deformation begins and can be written in the general form[2]

$$F(\sigma, \kappa) = 0 \tag{2.17}$$

where κ is the hardening parameter. On physical grounds, any yield criterion should be independent of the orientation of the coordinate system employed and therefore it should be a function of the three stress invariants only, J_1,J_2 and J_3. Experimental observations indicate that plastic deformation of metals is essentially independent of hydrostatic pressure. Consequently the yield function can only be of the form

$$F(J_2', J_3', \kappa) = 0 \tag{2.18}$$

in which J_2' and J_3' are the second and third invariants of the deviatoric stresses.

The situation is complicated by the fact that different classes of materials exhibit different elasto-plastic characteristics. In this thesis four commonly used yield criteria are introduced. The Tresca and Von Mises laws, which closely approximate metal plasticity behaviour, are considered and the Mohr-Coulomb and Drucker-Prager criteria, which are applicable to concrete, rocks and soils, are also presented.

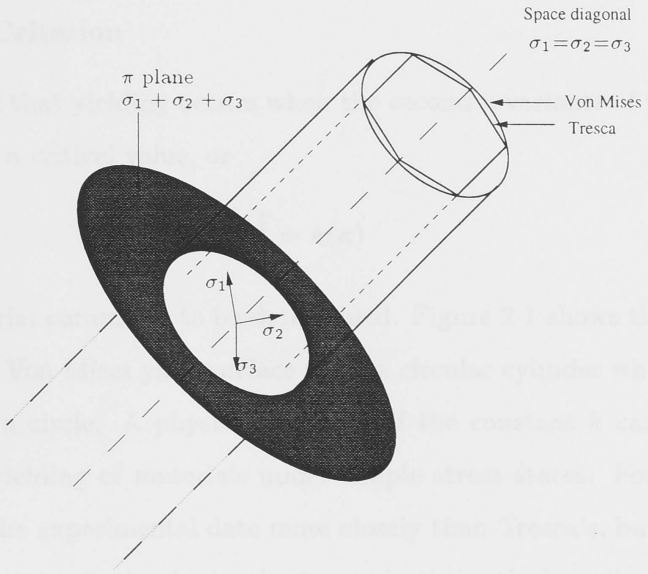

Figure 2-1: Geometrical representation of the Tresca and Von Mises yield surfaces in principal stress space[2]

Tresca Yield Criterion

The Tresca yield criterion states that yielding begins when the maximum shear stress reaches a certain value. If the principal stresses are σ_1, σ_2, σ_3 where $\sigma_1 \geq \sigma_2 \geq \sigma_3$ then yielding begins when

$$\sigma_1 - \sigma_3 = Y(\kappa) \tag{2.19}$$

where Y is a material parameter to be experimentally determined and which may be a function of the hardening parameter κ. By considering all other possible maximum shearing stress values it can be shown that this yield criterion may be represented in the $\sigma_1\sigma_2\sigma_3$ *stress space* by the surface of an infinitely long regular hexagonal cylinder as shown in Figure 2-1. The axis of the cylinder coincides with the space diagonal, defined by points $\sigma_1 = \sigma_2 = \sigma_3$, and since each normal section of the cylinder is identical, it is convenient to represent the yield surface geometrically by projecting it onto the so-called π plane, $\sigma_1 + \sigma_2 + \sigma_3 = 0$

Von Mises Yield Criterion

Von Mises suggested that yielding occurs when the second invariants of the deviatoric stresses, J_2', reaches a critical value, or

$$(J_2')^{\frac{1}{2}} = k(\kappa) \tag{2.20}$$

in which k is a material parameter to be determined. Figure 2-1 shows the geometrical interpretation of the Von Mises yield surface to be a circular cylinder whose projection onto the π plane is a circle. A physical meaning of the constant k can be obtained by considering the yielding of materials under simple stress states. For most metals Von Mises' law fits the experimental date more closely than Tresca's, but it frequently happens that the Tresca criterion is simpler to use in theoretical applications.

Mohr-Coulomb Yield Criterion

This is a generalisation of the Coulomb friction failure law defined by

$$\tau = c - \sigma_n \tan\phi \tag{2.21}$$

where τ is the magnitude of the shearing stress, σ_n is the normal stress, c is the cohesion and ϕ the angle of internal friction. Again, as for the Tresca criterion, the complete yield surface is obtained by considering all other stress combinations which can cause yielding (e.g. $\sigma_3 \leq \sigma_1 \leq \sigma_2$). In principal stress space this gives a conical yield surface whose normal section at any point is an irregular hexagon as shown in Figure 2-2. This criterion is applicable to concrete, rock and soil problems.

Drucker-Prager Yield Criterion

An approximation to the Mohr-Coulomb law was presented as a modification of the Von Mises yield criterion. The influence of a hydrostatic stress component on yielding was introduced by inclusion of an additional term in the Von Mises expression to give

$$\alpha J_1 + (J_2')^{\frac{1}{2}} = k' \tag{2.22}$$

This yield surface has the form of a circular cone. In order to make the Drucker-Prager circle coincide with the outer apices of the Mohr-Coulomb hexagon at any section, it

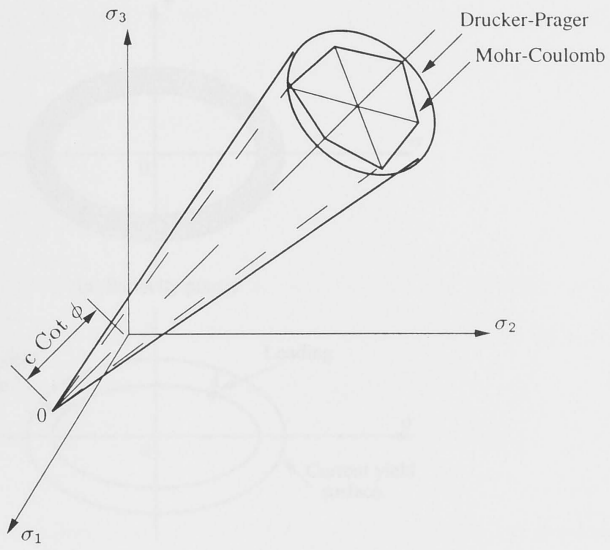

Figure 2-2: Geometrical representation of the Mohr-Coulomb and Drucker-Prager yield surfaces in principal stress space[2]

can be shown that

$$\alpha = \frac{2\sin\phi}{\sqrt{(3)}(3-\sin\phi)}, \quad k' = \frac{6ccos\phi}{\sqrt{(3)}(3-\sin\phi)} \tag{2.23}$$

Coincidence with the inner apices of the Mohr-Coulomb hexagon is provided by

$$\alpha = \frac{2\sin\phi}{\sqrt{(3)}(3+\sin\phi)}, \quad k' = \frac{6ccos\phi}{\sqrt{(3)}(3+\sin\phi)} \tag{2.24}$$

However, the approximation given by either the inner or outer cone to the true failure surface can be poor for certain stress combinations[2].

2.2.2 Work or Strain Hardening

After initial yielding, the stress level at which further plastic deformation occurs may be dependent on the current degree of plastic straining. Such a phenomenon is termed work hardening or strain hardening. Thus the yield surface will vary at each stage of the plastic deformation, with the subsequent yield surfaces being dependent on the plastic strains in some way. Some alternative models which describe strain hardening in a material are illustrated in Figure 2-3. A perfectly plastic material is shown in Figure 2-3(a) where the yield stress level does not depend in any way on the degree of plastification. If the subsequent yield surfaces are a uniform expansion of the original yield curve, without translation, as shown in Figure 2-3(b) the strain hardening model

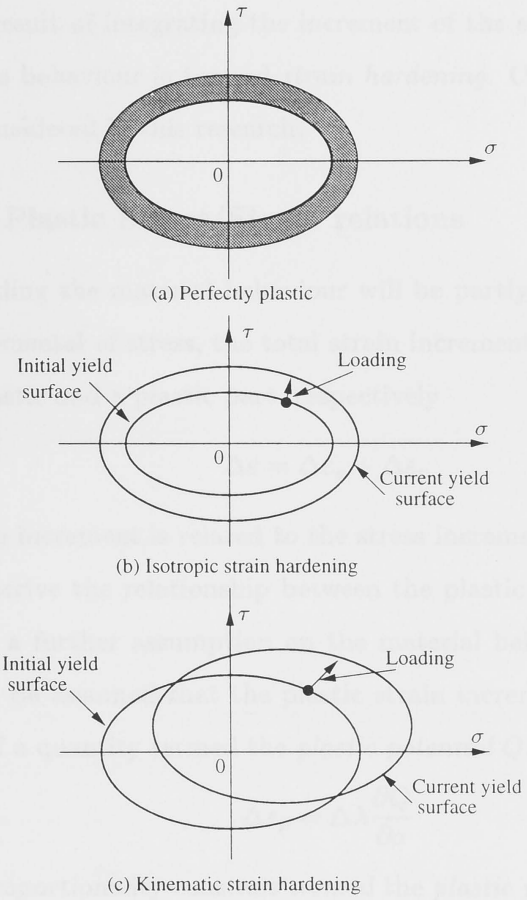

Figure 2-3: Mathematical models for representation of strain hardening behaviour[2]

is said to be isotropic. On the other hand if the subsequent yield surfaces preserve their shape and orientation but translate in th stress space as a rigid body as shown in Figure 2-3(c), kinematic hardening is said to take place. Such a hardening model gives rise to the experimentally observed Bauschinger effect on cyclic loading.

The progressive development of the yield surface can be defined by relating the yield stress Y to the plastic deformation by mean of the hardening parameter κ. This can be done in two ways. Firstly the degree of *work hardening* can be postulated to be a function of the total plastic work, W_p, only. Then,

$$\kappa = f(W_p) \tag{2.25}$$

Alternatively κ can be related to a measure of the total plastic deformation termed the *effective plastic strain*. Then the hardening parameter is assumed to be defined as

$$\kappa = \bar{\varepsilon}_p \tag{2.26}$$

where $\bar{\varepsilon}_p$ is the result of integrating the increment of the effective strain $d\bar{\varepsilon}_p$ over the strain path. This behaviour is termed *strain hardening.* Only an isotropic hardening model will be considered in this research.

2.2.3 Elasto-Plastic Stress/Strain relations

After initial yielding the material behaviour will be partly elastic and partly plastic. During any incremental of stress, the total strain increment may be additively decomposed into an elastic and a plastic part, respectively

$$\Delta\varepsilon = \Delta\varepsilon_e + \Delta\varepsilon_p \tag{2.27}$$

The elastic strain increment is related to the stress increment by Equation 2.13.

In order to derive the relationship between the plastic strain component and the stress increment a further assumption on the material behaviour must be made. In particular it will be assumed that the plastic strain increment is proportional to the stress gradient of a quantity termed the *plastic potential* Q, so that

$$\Delta\varepsilon_p = \Delta\lambda\frac{\partial Q}{\partial\sigma} \tag{2.28}$$

where $\Delta\lambda$ is a proportionality constant termed the *plastic multiplier.* Equation 2.28 is termed the *flow rule* since it governs the plastic flow after yielding. The potential Q must be a function of J_2' and J_3' but as yet it cannot be determined in its most general form. However the relation $F \equiv Q$ has a special significance in the mathematical theory of plasticity, since for this case certain variational principles and uniqueness theorems can be formulated. Such assumption give rise to an *associated plasticity.*

When plastic yielding is occuring the stresses are on the yield surface given by Equation 2.17. Differentiating this we can therefore write

$$\Delta F = \frac{\partial F}{\partial\sigma_1}\Delta\sigma_1 + \frac{\partial F}{\partial\sigma_2}\Delta\sigma_2 + ... + \frac{\partial F}{\partial\kappa}\Delta\kappa = 0 \tag{2.29}$$

By using Equation 2.27-2.29, we obtain, after some transformation, the complete elasto-plastic incremental stress-strain relation to be

$$\Delta\sigma = D_{ep}\Delta\varepsilon \tag{2.30}$$

where

$$D_{ep} = D - D\left\{\frac{\partial Q}{\partial\sigma}\right\}\left\{\frac{\partial F}{\partial\sigma}\right\}^T D\left[H' + \left\{\frac{\partial F}{\partial\sigma}\right\}^T D\left\{\frac{\partial Q}{\partial\sigma}\right\}\right]^{-1} \tag{2.31}$$

If the flow rule is assumed to be associated, then we can simply write D_{ep} as

$$D_{ep} = D - \frac{d_D {d_D}^T}{H' + {d_D}^T a}; \quad d_D = Da \tag{2.32}$$

and

$$a^T = \frac{\partial F}{\partial \sigma} = \left[\frac{\partial F}{\partial \sigma_x}, \frac{\partial F}{\partial \sigma_y}, \frac{\partial F}{\partial \sigma_z}, \frac{\partial F}{\partial \tau_{xy}}, \frac{\partial F}{\partial \tau_{yz}}, \frac{\partial F}{\partial \tau_{zx}}\right], \tag{2.33}$$

Clearly for ideal plasticity with no hardening, H' is simply zero. When a linear work hardening is considered, H' is obtained to be the local slope of the uniaxial stress/plastic strain curve, which is constant and can be determined experimentally[2].

2.2.4 Alternative Form of the Yield Criterion for Numerical Computation

For numerical computation it is convenient to rewrite the yield function in terms of alternative stress invariants. The main advantage of this formulation is that it permits the computer coding of the yield function and flow rule in a general form and necessitates only the specification of three constants for any individual criterion. Normally a parameter θ (Lode angle) is used in such definition. By noting the cyclic nature of $\sin(3\theta + 2n\pi)$ we have immediately the three possible values of $\sin\theta$ which define the three principal stresses. In [2], the total principal stresses are expressed as

$$\begin{Bmatrix} \sigma_1 \\ \sigma_2 \\ \sigma_3 \end{Bmatrix} = \frac{2(J_2')^{\frac{1}{2}}}{\sqrt{3}} \begin{Bmatrix} \sin\left(\theta + \frac{2\pi}{3}\right) \\ \sin\theta \\ \sin\left(\theta + \frac{4\pi}{3}\right) \end{Bmatrix} + \frac{J_1}{3} \begin{Bmatrix} 1 \\ 1 \\ 1 \end{Bmatrix} \tag{2.34}$$

with $\sigma_1 > \sigma_2 > \sigma_3$ and $-\pi/6 \leq \theta \leq \pi/6$. The four yield criteria considered in section 2.2.1 can now be rewritten in terms of J_1, J_2' and θ as follows.

The Tresca Yield Criterion

Substitute for σ_1 and σ_3 from Equation 2.34 into Equation 2.19 gives

$$\frac{2}{\sqrt{3}}(J_2')^{\frac{1}{2}}\left[\sin\left(\theta + \frac{2\pi}{3}\right) - \sin\left(\theta + \frac{4\pi}{3}\right)\right] = Y(\kappa)$$

or expanding we have

$$2(J_2')^{\frac{1}{2}}\cos\theta = Y(\kappa) = \sigma_Y(\kappa) \tag{2.35}$$

Table 2.1: Effective stress and uniaxial yield stress levels for the yield criteria included in the elasto-plastic computer code

Equation No.	Yield criterion	Stress level (effective stress)	Uniaxial (or equivalent yield stress)
(2.35)	Tresca	$2(J_2')^{\frac{1}{2}}\cos\theta$	σ_Y
(2.36)	Von Mises	$\sqrt{3}(J_2')^{\frac{1}{2}}$	σ_Y
(2.37)	Mohr-Coulomb	$\frac{1}{3}J_1\sin\phi + (J_2')^{1/2}$ $\times(\cos\theta - \sin\theta\sin\phi/\sqrt{3})$	$c\cos\phi$
(2.38)	Drucker-Prager	$\alpha J_1 + (J_2')^{\frac{1}{2}}$	k'

The Von Mises Yield Criterion

There is no change in this case since this yield function depends on J_2' only. We rewrite Equation 2.20

$$\sqrt{3}(J_2')^{\frac{1}{2}} = \sigma_Y(\kappa) \tag{2.36}$$

The Mohr-Coulomb Yield Criterion

Substitute for σ_1 and σ_3 from Equation 2.34 into Equation 2.21 results in

$$\frac{1}{3}J_1\sin\phi + (J_2')^{1/2}\left(\cos\theta - \frac{1}{\sqrt{3}}\right) = c\cos\phi \tag{2.37}$$

The Drucker-Prager Yield Criterion

There is no change for this criterion and we can write directly from Equation 2.22 that

$$\alpha J_1 + (J_2')^{\frac{1}{2}} = k' \tag{2.38}$$

where α and k' are defined in Equation 2.23 or 2.24. The four yield criteria are summarised in Table 2.1

In order to calculate the D_{ep} matrix in Equation 2.32, we also require to express the flow vector a in a form suitable for numerical computation. We can always write

$$a^T = \frac{\partial F}{\partial \sigma} = \frac{\partial F}{\partial J_1}\frac{\partial J_1}{\partial \sigma} + \frac{\partial F}{\partial (J_2')^{1/2}}\frac{\partial (J_2')^{1/2}}{\partial \sigma} + \frac{\partial F}{\partial \theta}\frac{\partial \theta}{\partial \sigma} \tag{2.39}$$

we can then write

$$a = C_1 a_1 + C_2 a_2 + C_3 a_3 \tag{2.40}$$

Table 2.2: Constants defining the yield surface in a form suitable for numerical analysis

Yield Criterion	C_1	C_2	C_3
Tresca	0	$2\cos\theta(1+\tan\theta\tan 3\theta)$	$\dfrac{\sqrt{3}\ \sin\theta}{J_2'\ \cos 3\theta}$
Von Mises	0	$\sqrt{3}$	0
Mohr-Coulomb	$\frac{1}{3}\sin\theta$	$\cos\theta[(1+\tan\theta\tan 3\theta)$ $+\sin\theta(\tan 3\theta-\tan\theta)/\sqrt{3}]$	$\dfrac{(\sqrt{3}\sin\theta+\cos\theta\sin\phi)}{(2J_2'\cos 3\theta)}$
Drucker-Prager	α	1.0	0

where

$$a_1{}^T = \frac{\partial J_1}{\partial \sigma}, \quad a_2{}^T = \frac{\partial (J_2')^{1/2}}{\partial \sigma}, \quad a_3{}^T = \frac{\partial J_3}{\partial \sigma} \tag{2.41}$$

and

$$\begin{aligned} C_1 &= \frac{\partial F}{\partial J_1}, \quad C_2 = \left(\frac{\partial F}{\partial (J_2')^{1/2}} - \frac{\tan 3\theta}{(J_2')^{1/2}} \frac{\partial F}{\partial \theta} \right) \\ C_3 &= \frac{-\sqrt{3}}{2\cos 3\theta} \frac{1}{(J_2')^{3/2}} \frac{\partial F}{\partial \theta} \end{aligned} \tag{2.42}$$

Only the constants C_1, C_2 and C_3 are then necessary to define the yield surface. Thus we can achieve a simplicity of programming as only these three constants have to be varied between one yield surface and another. The constants C_i are given in Table 2.2 for the four yield criteria considered.

2.2.5 Determination of Initial Yielding State

During the application of an increment of load an element, or part of an element, may yield. All stress and strain quantities are monitored at each Gaussian integration point and therefore we can determine whether or not plastic deformation has occurred at such points. Consequently an element can behave partly elastically and partly elasto-plastically if some, but not all, Gauss points indicate plastic yielding. For any load increment it is necessary to determine what proportion is elastic and which part produces plastic deformation and then adjust the stress and strain terms until the yield criterion and the constitutive laws are satisfied.

Within a particular load increment, the displacements increments Δu^r can be determined by solving a linear system of equations. The strain increments at an integration point may be computed from the strain-displacement relations according to

$$\Delta\varepsilon^r = B\Delta u^r \tag{2.43}$$

where r denotes the r^{th} iteration of current load step. Once the strains have been determined, the elastic stress increments (i.e. the trial stress) may be calculated using Hooke's law:

$$\Delta{\sigma_e}^r = D\Delta\varepsilon^r \tag{2.44}$$

and a trial stress state is obtained through

$${\sigma_e}^r = \sigma^{r-1} + \Delta{\sigma_e}^r \tag{2.45}$$

where the subscript e denotes that we are assuming elastic behaviour. The trial stress ${\sigma_e}^r$ is then tested in Equation 2.17. If $F < 0$, then the elasticity assumption is taken to be valid, and ${\sigma_e}^r$ is considered as the new stress state. Otherwise, the strain increment is partly in an elastic path and partly in a plastic path. In order to determine the portion of the stress which cause purely plastic yielding, we need to find a scalar α such that

$$F(\sigma^r, H) = 0$$

where

$$\sigma^r = \sigma^{r-1} + (1-\alpha)\Delta{\sigma_e}^r$$

A variety of schemes are available for determining scalar α. In [2], Sloan used a Newton-Raphson iteration to compute α. It should be noted that performing iterations in the integration scheme may lead to better results but the procedures often fails to converge. In this thesis, a linear interpolation method is used. we rewrite Equation 2.17 in the following form

$$F(\sigma, H) = \bar{\sigma} - Y(H) = 0 \tag{2.46}$$

where $\bar{\sigma}$ and $Y(H)$ denote the effective stress and the isotropic hardening function, respectively. For an isotropic strain-hardening model, which will be employed later in

this thesis, the hardening parameter is assumed to be related to some measure of the plastic strains. So we can write $Y(H)$ in the following form:

$$Y(H) = Y(\bar{\varepsilon}_p) = {\sigma_Y}^o + H'\bar{\varepsilon}_p \tag{2.47}$$

where ${\sigma_Y}^o$ denotes the uniaxial yield stress. Then α can be obtained by

$$\alpha = \frac{F({\sigma_e}^r, Y^{r-1})}{F({\sigma_e}^r, Y^{r-1}) - F(\sigma^{r-1}, Y^{r-1})} = \frac{\bar{\sigma}_e^r - Y^{r-1}}{\bar{\sigma}_e^r - \bar{\sigma}^{r-1}} \tag{2.48}$$

where $Y^{r-1} = {\sigma_Y}^o + H'\bar{\varepsilon}_p^{r-1}$. After determining the portion of the stress increments which cause purely plastic yielding, we must reduce the excess stress onto the yield surface until yield criterion and the constitutive law are satisfied.

2.3 Integration Algorithms for Stress-Strain Relations

Once the stresses at the onset of initial yielding have been computed, the integration of stress-strain relations requires the solution of the initial value problem given by

$$\frac{d\sigma}{dT} = D_{ep}\Delta\varepsilon, \quad T \in [0,1] \tag{2.49}$$

in which $\sigma|_{T=0}$ defines the stress state which already satisfy the yield criterion, and $\sigma|_{T=1}$ defines the stress at an end of load increment or iteration. Both the conventional integration algorithm and a substepping scheme are studied in this research.

2.3.1 Conventional Method

The crudest method for solving the system of differential equations defined by 2.49 is the Euler algorithm. Since the Euler scheme is accurate only for very small time steps, it is always necessary to divide the whole integration process into a number of smaller substeps. The conventional method is introduced in the following.

1. Enter with the stress σ^{r-1} and hardening modulus H', together with the displacements increments for the current load step Δu^r and the error tolerance TOL.

2. Compute the strain increment $\Delta\varepsilon^r$ and the trial stress increment $\Delta{\sigma_e}^r$ using Equation 2.43 and 2.44. If $F(\sigma^{r-1} + \Delta{\sigma_e}^r) \leq 0$, go to step 6.

3. If $F(\sigma^{r-1}) < 0$ and $F(\sigma^{r-1}+\Delta\sigma_e{}^r) > 0$, which means the Gauss point has yielded during application of load corresponding to this iteration as shown in Figure 2-5, compute the portion of $\Delta\sigma_e{}^r$ that cause purely plastic deformation (i.e. compute the α factor as described in the section on initial yielding). If the point underwent plastic yielding in the previous load step or iteration and $F(\sigma^{r-1}+\Delta\sigma_e{}^r) > 0$, the Gauss point had yielded previously and the stress is still increasing. Therefore all the stress must be reduced to the yield surface as indicated in Figure 2-4, then set $\alpha = 1$.

4. Compute the portion of $\Delta\sigma_e{}^r$ that causes plastic deformation according to $\Delta\sigma_e = \alpha\sigma_e{}^r$. Then update the stresses at the onset of plastic yielding according to $\sigma \leftarrow \sigma^{r-1} + (1-\alpha)\Delta\sigma_e{}^r$.

5. The remaining portion of stress, $\alpha\Delta\sigma_e^r$ must be effectively eliminated in some way. The point A (Figure 2-5) must be brought onto the yield surface by allowing plastic deformation to occur. Physically this can be described as follows. On loading from point C, the stress point moves elastically until the yield surface is met at B. Elastic behaviour beyond this point would result in a final stress state defined by point A. However in order to satisfy the yield surface and consequently the stress point can only traverse the surface until both equilibrium conditions

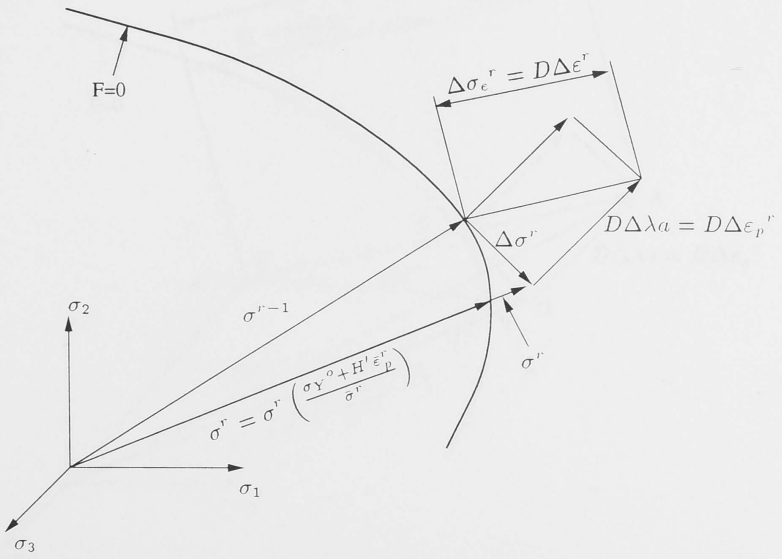

Figure 2-4: Incremental stress changes in an already yielded point in an elasto-plastic continuum.

and the constitutive relation are satisfied. Normally we have

$$\sigma^r = \sigma^{r-1} + \Delta\sigma_e^r - \Delta\lambda d_D \tag{2.50}$$

which gives the total stresses σ^r satisfying elasto-plastic conditions when the stresses are incremented from σ^{r-1}. Equation 2.50 is illustrated vectorially in Figure 2-4 and 2-5. It is seen that if a finite sized stress increment is taken, the final stress point D, corresponding to σ^r, may depart from the yield surface. This discrepancy can be practically eliminated by ensuring that the load increments considered in solution are sufficiently small. However the point D can be reduced to the yield surface by simply scaling the vector σ^r. Denoting the effective stress, given by Col.3, Table 2.1, due to stress σ^r as $\bar{\sigma}^r$ and noting that this value should coincide with $\sigma_Y = \sigma_Y{}^o + H'\bar{\epsilon}_p^r$ if the point D lies on the yield surface, the appropriate scaling factor is readily seen to be

$$\sigma^r = \sigma^r \left(\frac{\sigma_Y{}^o + H'\bar{\epsilon}_p^r}{\bar{\sigma}^r} \right) \tag{2.51}$$

This represents a scaling of the vector σ^r which implies that the individual stress components are proportionally reduced. The normality condition for the plastic

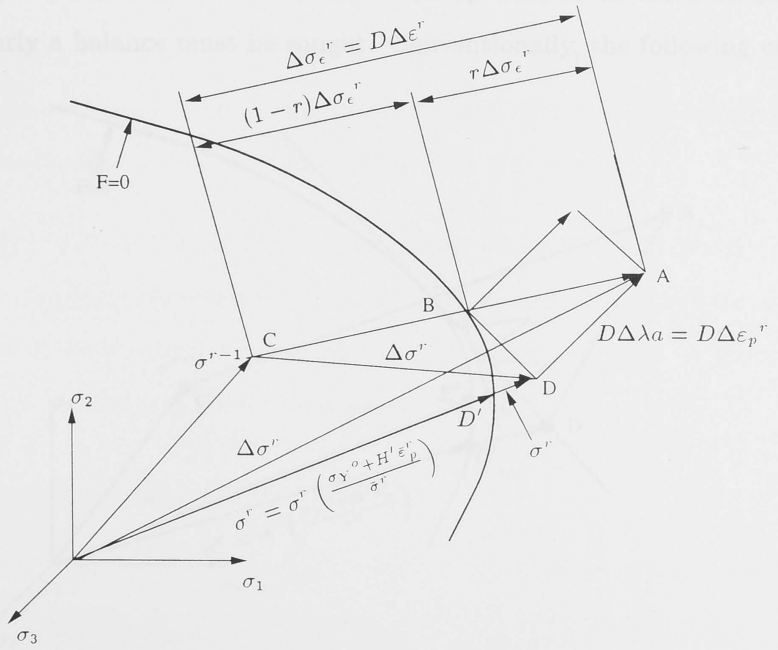

Figure 2-5: Incremental stress changes at a point in an elasto-plastic continuum at initial yield.

strain increment is evident from Figure 2-4 and 2-5 since $D\Delta\lambda a = D\Delta\varepsilon_p$.

If relatively large load increment sizes are to be permitted, the process described above can lead to an inaccurate prediction of the final point D on the yield surface if the stress point is in the vicinity of a region of large curvature of the yield surface. This is illustrated on Figure 2-6 where the process of reducing the elastic stress to the yield surface is shown to end in the stress point D which is then scaled down to the yield surface to give point D'. Greater accuracy can be achieved by relaxing the excess stress to the yield surface in several stages. Figure 2-6 shows the case where the excess stress is divided into three equal parts and each increment reduced to the yield surface in turn. After the three reduction cycles to the stress point E the drift away from the yield surface can be corrected by simple scaling to give the final stress point E'. It is seen that the final points D' and E' can be significantly different. An additional refinement which can be introduced is to scale the stress point to the yield surface after the reduction process for each cycle and not only after the final cycle as shown in Figure 2-6. Obviously the greater the number of steps into which the excess stress AB is divided, the greater the accuracy. However the computation for each step is relatively expensive since the vectors a and d_D have to be calculated at each stage. Clearly a balance must be sought. Conventionally, the following criterion

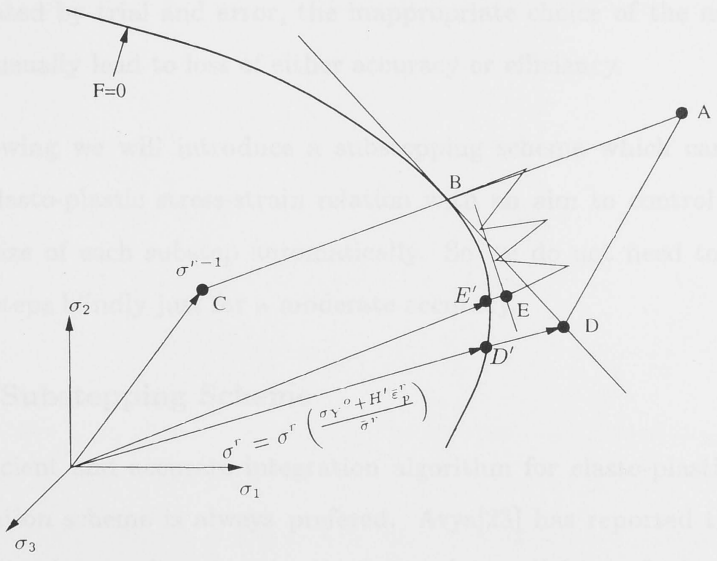

Figure 2-6: Refined process for reducing a stress point to the yield surface.

is adopted. The excess stress $\alpha\Delta\sigma_e^r$ is divided into m parts where m is given by the nearest integer which is less than

$$(\frac{\bar{\sigma}_e^r - \sigma_Y}{\sigma_Y{}^0})8 + 1 \tag{2.52}$$

where $\bar{\sigma}_e^r - \sigma_Y$ gives a measure of the excess stress AB and σ_Y^0 is the initial uniaxial yield stress in Col.4, Table 2.1 before the onset of work hardening.

6. For elastic Gauss points only calculate $\sigma^r = \sigma^{r-1} + d\sigma_e^r$.

7. Finally, calculate the equivalent nodal forces from the element stresses according to

$$(f^{(e)})^r = \int_{\Omega^{(e)}} B^T \sigma^r d\Omega \tag{2.53}$$

Although this method has been used widely in the finite element codes, it has the following disadvantages:

1. If the correction-step is applied after each substep, the computational time will increase drastically. However, if it is done at the end of integration, it does not significantly affect the accuracy[4].

2. Since the number of substeps is usually determined by an empirical rule which is formulated by trial and error, the inappropriate choice of the number of the substeps usually lead to loss of either accuracy or efficiency.

In the following we will introduce a substepping scheme which can be used to integrate the elasto-plastic stress-strain relation with an aim to control the error by adjusting the size of each substep automatically. So we do not need to increase the number of substeps blindly just for a moderate accuracy.

2.3.2 New Substepping Scheme

To develop efficient and accurate integration algorithm for elasto-plastic models, an explicit integration scheme is always prefered. Arya[23] has reported that a simple, self-adaptive time integration strategy involving the explicit method can work very successfully if the step size is properly controlled. In [4], it is shown that methods of

high order can be formulated for elasto-plastic problems, which are much more efficient than the first order algorithms used up to the present. It should be noted that the fifth-order Runge-Kutta-England method, which is used in Sloan[8], has been found to be highly desirable where quite high accuracy is required. In this thesis, fourth-order Runge-Kutta method is used since it not only provide sufficient accuracy, but also saves computational time. Since the substepping scheme controls the error by decreasing the step size, it definitely involves a large number of substeps. As such, the cumulative effect of the per-step roundoff errors and their magnification in calculating subsequent substeps must be minimized. In this thesis, we will employ Gill's fourth-order Runge-Kutta method which is known for its advantage of minimizing the roundoff error[30].

For the third-order Runge-Kutta method, the solution to Equation 2.49 at the end of a substep ΔT_k is given by

$$\sigma_{k+1} = \sigma_k + \frac{1}{6}(\Delta\sigma_1 + 4\Delta\sigma_2 + \Delta\sigma_3) \tag{2.54}$$

where

$$\Delta\sigma_i = D_{ep}(\sigma_i)\Delta\varepsilon_k \tag{2.55}$$

and $\Delta\varepsilon_k = \Delta T_k \Delta\varepsilon$. A more accurate estimate of σ_{k+1} may be obtained from Gill's fourth-order Runge-Kutta method and is given by

$$\hat{\sigma}_{k+1} = \sigma_k + \frac{1}{6}\left[\Delta\sigma_1 + (2-\sqrt{2})\Delta\sigma_2 + (2+\sqrt{2})\Delta\sigma_3 + \Delta\sigma_4\right] \tag{2.56}$$

where

$$\begin{aligned}
\sigma_1 &= \sigma_k \\
\sigma_2 &= \sigma_k + \frac{1}{2}\Delta\sigma_1 \\
\sigma_3 &= \sigma_k + (\sqrt{2}-1)\frac{\Delta\sigma_1}{2} + (2-\sqrt{2})\frac{\Delta\sigma_2}{2} \\
\sigma_4 &= \sigma_k - \frac{\sqrt{2}}{2}\Delta\sigma_2 + (1+\frac{\sqrt{2}}{2})\Delta\sigma_3
\end{aligned} \tag{2.57}$$

To estimate the local truncation error in the solution, an 'imbedding' technique is employed[23]. In this technique the local error is defined as the difference in the solution obtained by using two methods of different orders. Now for a given strain increment $\Delta\varepsilon$, the third-order Runge-Kutta method has the per-step error of order $O(\Delta T)^4$, whereas the per-step error in the fourth-order method is of order $O(\Delta T)^5$. Thus, subtracting

Equation 2.54 from Equation 2.56, we obtain an estimate of the local truncation error in σ_{k+1} according to

$$E_{k+1} = \frac{1}{6}\left[-(2+\sqrt{2})\Delta\sigma_2 + (1+\sqrt{2})\Delta\sigma_3 + \Delta\sigma_4)\right] \tag{2.58}$$

As an estimate for the local error in the substep from T_k to $T_{k+1} = T_k + \Delta T_k$, we define the relative error for this substep as

$$R_{k+1} = \frac{\| E_{k+1} \|}{\| \hat{\sigma}_{k+1} \|} \tag{2.59}$$

Then R_{k+1} is compared with some prescribed tolerance TOL_{sub} and the step is accepted if $R_{k+1} \leq TOL_{sub}$, and rejected otherwise. Furthermore, the value of R_{k+1} allows us to make an estimate for the asymptotically optimal stepsize:

$$\Delta T_{k+1} = \Delta T_k \sqrt[4]{TOL_{sub}/R_{k+1}}$$

In case of rejection, ΔT_{k+1} is used instead of ΔT_k; in case of acceptance we use ΔT_{k+1} to continue the integration. In the first case, it is not prudent to use ΔT_{k+1} without introducing a safety margin. Otherwise, the number of rejections will be larger than necessary. In order to give the code some robustness, we actually implemented

$$\Delta T_{k+1} = \Delta T_k \cdot min\left\{2, max\left\{0.1, 0.9\sqrt[4]{TOL_{sub}/R_{k+1}}\right\}\right\} \tag{2.60}$$

The constants 2 and 0.1 in this expression serve to prevent an abrupt change in the substep size, and the safety factor 0.9 is added to increase the probability that next substep will be accepted[70]. By controlling the local relative error for each substep, we aim to control the global relative error in the overall solution.

The Runge-kutta algorithm, which incorporates error control and a variable step size for each integration point, may be summarized as follows:

1. Enter with the stress σ^{r-1} and hardening modulus H', together with the displacements increments for the current load step Δu^r and the error tolerance TOL.

2. Compute the strain increment $\Delta\varepsilon^r$ and the trial stress increment $\Delta\sigma_e{}^r$ using Equation2.43 and 2.44. If $F(\sigma^{r-1} + \Delta\sigma_e{}^r) \leq 0$, set $\sigma \leftarrow \sigma^{r-1} + \Delta\sigma_e{}^r$ and go to step 12.

3. If $F(\sigma^{r-1}) < 0$ and $F(\sigma^{r-1} + \Delta\sigma_e{}^r) > 0$, compute the portion of $\Delta\sigma_e{}^r$ that cause purely plastic deformation (i.e. scalar parameter α). If the point underwent plastic yielding in the previous load step or iteration and $F(\sigma^{r-1} + \Delta\sigma_e{}^r) > 0$, then set $\alpha = 1$.

4. Compute the portion of $\Delta\sigma_e{}^r$ that causes plastic deformation according to $\Delta\sigma_e = \alpha\sigma_e{}^r$. Then update the stresses at the onset of plastic yielding according to $\sigma \leftarrow \sigma^{r-1} + (1-\alpha)\Delta\sigma_e{}^r$.

5. Set $T = 0$ and $\Delta T = 1$.

6. While $T < 1$, do step 7 to 11.

7. Compute $\Delta\sigma_i$ for i=1,4 according to Equation 2.55 and Equation 2.57.

8. Compute an estimate of the local truncation error for the substep ΔT according to Equation 2.71 and compute the new stresses using Equation 2.56.

9. Determine the relative error for the substep ΔT from Equation 2.59.

10. If $R > TOL$, then go to step 11. Else, this substep is accepted so update T and the stresses according to

$$T \leftarrow T + \Delta T, \quad \sigma = \hat{\sigma}$$

Then extrapolate to obtain the size of the next substep using Equation 2.60. Before returning to step 6, check that the integration does not proceed beyond T = 1 by setting

$$\Delta T \leftarrow min\{\Delta T, 1 - T\}$$

11. This substep has failed, so extrapolate to obtain a smaller substep by using Equation 2.60. Then return to step 6.

12. Exit with stresses at time T = 1 given by $\sigma^r = \sigma$.

Note that in each substep, four evaluations of the D_{ep} matrix are required for each substep instead of evaluating elasto-plastic matrix six times by using fifth-order Runge-Kutta method. If $\Delta\lambda$ is found to be less than zero during a substep, then $\Delta\lambda$ is set to be zero.

In the following, Modified Euler scheme and Runge-Kutta-England scheme will be chosen for comparison. So it is necessary to have a brief introduction of them.

Modified Euler scheme

In the modified Euler algorithm, the error in the stress is computed by comparing the results from the first-order Euler method and the modified Euler scheme, respectively

$$\sigma_{k+1} = \sigma_k + \Delta\sigma_1 \tag{2.61}$$

$$\hat{\sigma}_{k+1} = \sigma_k + \frac{1}{2}(\Delta\sigma_1 + \Delta\sigma_2) \tag{2.62}$$

where

$$\begin{aligned} \sigma_1 &= \sigma_k \\ \sigma_2 &= \sigma_k + \Delta\sigma_1 \end{aligned} \tag{2.63}$$

Based on the Equation (2.61) and (2.62), we can obtain the estimate of the local truncation error in σ_{k+1}

$$E_{k+1} = \frac{1}{2}(-\Delta\sigma_1 + \Delta\sigma_2) \tag{2.64}$$

The only part which is different from the Runge-Kutta method is the computation of the size of next substep ΔT_{k+1}. In modified Euler scheme, ΔT_{k+1} is determined according to

$$\Delta T_{k+1} = \Delta T_k \cdot min\left\{2, max\left\{0.1, 0.8\sqrt[2]{TOL_{sub}/R_{k+1}}\right\}\right\} \tag{2.65}$$

Runge-Kutta-England scheme

In this algorithm, fourth and fifth order Runge-Kutta methods are employed. Similar to the modified Euler scheme, the error estimate is obtained by comparing the estimated stress increments which result from the fourth and fifth order Runge-Kutta-England, respectively,

$$\sigma_{k+1} = \sigma_k + \frac{1}{6}(\Delta\sigma_1 + 4\Delta\sigma_3 + \Delta\sigma_4) \tag{2.66}$$

and

$$\hat{\sigma}_{k+1} = \sigma_k + \frac{1}{336}(14\Delta\sigma_1 + 35\Delta\sigma_4 + 162\Delta\sigma_5 + 125\Delta\sigma_6) \tag{2.67}$$

in which

$$\begin{aligned}
\sigma_1 &= \sigma_k \\
\sigma_2 &= \sigma_k + \frac{1}{2}\Delta\sigma_1 \\
\sigma_3 &= \sigma_k + \frac{1}{4}(\Delta\sigma_1 + \Delta\sigma_2) \qquad (2.68) \\
\sigma_4 &= \sigma_k - \Delta\sigma_2 + 2\Delta\sigma_3 \\
\sigma_5 &= \sigma_k + \frac{1}{27}(7\Delta\sigma_1 + 10\Delta\sigma_2 + \Delta\sigma_4) \qquad (2.69) \\
\sigma_6 &= \sigma_k + \frac{1}{625}(28\Delta\sigma_1 - 125\Delta\sigma_2 + 546\Delta\sigma_3 + 54\Delta\sigma_4 - 378\Delta\sigma_5) \qquad (2.70)
\end{aligned}$$

and the estimate of the local truncation error is obtained according to

$$E_{k+1} = \frac{1}{336}(-42\Delta\sigma_1 - 224\Delta\sigma_3 - 21\Delta\sigma_4 + 162\Delta\sigma_5 + 125\Delta\sigma_6) \qquad (2.71)$$

The next time step ΔT_{k+1} is computed from

$$\Delta T_{k+1} = \Delta T_k \cdot min\left\{2, max\left\{0.1, 0.8\sqrt[5]{TOL/R_{k+1}}\right\}\right\} \qquad (2.72)$$

The detailed introduction of these two integration schemes can be found in [8]. For a single substep, the modified Euler scheme and Runge-Kutta-England scheme require two and six evaluations of D_{ep}, respectively. In the following, the performance of the modified Euler scheme will be compared with the Gill's fourth-order Runge-Kutta scheme developed in this research work.

2.4 Overall Solution Methods

2.4.1 Iterative Methods

Almost without exception, the solution to non-linear finite element problems requires iteration. Since many models exhibit a convoluted load deformation response, a single iterative method may mot lead to a solution for the entire load history. The most common approach to the solution of nonlinear problems has been to apply loading, through specified force and /or displacements, in distinct steps. Within each of the steps, iterations are performed until equilibrium is satisfied based on a convergence criterion. Using a Taylor's series expansion, the non-linear finite element problem can be reduced to the following iterative matrix equation:

$$K\Delta U^{(i)} = R - F^{(i-1)} \qquad (2.73)$$

where K is the global stiffness matrix, $\Delta U^{(i)}$ is the incremental nodal displacement vector for iteration i, R is the vector of externally applied nodal loads for the present load step and $F^{(i-1)}$ is the vector of internal nodal forces which equilibrate the element stresses of iteration $i-1$. The solution of Equation 2.73 for the incremental displacement vector requires considerable computational effort. Many techniques have been developed in an attempt to minimize this effort. Three commonly used procedures are listed as follows:

1. The Newton-Raphson (N-R) method;
2. The Modified Newton-Raphson (M-N-R) method;
3. The Quasi-Newton-Raphson (Q-N-R) method.

In the standard N-R method, the tangential stiffness matrix is recomputed at every iteration. It has been found that this method exhibits the fastest convergence rate. The M-N-R and Q-N-R methods can greatly reduce the computational effort by keeping the stiffness matrix constant for some or all of the iterations within one load increment or even for several load increments. Unfortunately, convergence of these methods is much slower than the N-R procedure. Matthies and Strang[24] have suggested that even more drastic modification to the standard N-R method is possible by keeping the elastic matrix constant used in the first iteration of first load increment throughout the overall solution. However, for large three-dimensional problems, the time saved by keeping the stiffness matrix constant can be offset by the repeated solution of system of equations due to the low convergence rate. So in the present work, only the standard N-R method is used.

2.4.2 Load Increment Control

In this thesis, we applied the load incrementally according to the load factor specified as input. The applied loading is accumulative so that if load factor is input as 0.5, 0.3 and 0.1 for the first three increments, the total load acting on the structure during the third load increment is 90 percent of the total applied load. This method of load factoring allows both equal and unequal load increments.

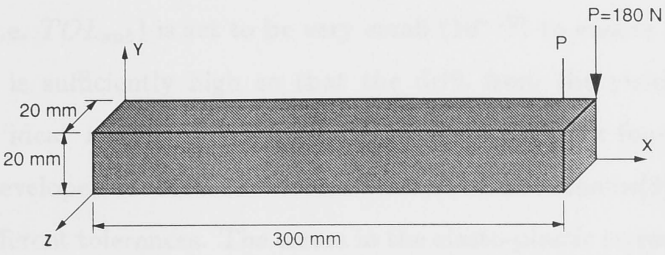

Figure 2-7: A typical cantilever beam

2.4.3 Convergence Criteria

To finish the iterative process in N-R method, a convergence criterion must be defined. The convergence criterion employed in this research compares the norm of residual forces against the norm of the applied forces, that is

$$\frac{\|\psi^r\|}{\|f^r\|} < TOL_{ep} \tag{2.74}$$

where

$$\|\psi^r\| = \left[\sum_{i=1}^{N} (f_i{}^r - g_i{}^r)^2\right]^{1/2}$$

$$\|f^r\| = \left[\sum_{i=1}^{N} (f_i{}^r)^2\right]^{1/2}$$

in which N is the total number of degrees of freedom in the problem and r denotes the iteration number. f_i and g_i are components of the applied forces and the equivalent force of r^{th} iteration at degree of freedom i.

2.5 Performance Analysis of Substepping Schemes

In order to illustrate the performance of the proposed algorithm in the finite element analysis, a computer program was developed and has been applied to a typical three dimensional cantilever beam (Figure 2-7). An associated flow rule is assumed. The Von Mises yield surface is employed and nodal force P is set to be 2.0 KN. The analyses were carried out in 10 equal load increments. To test the influence of different tolerances on the accuracy, an "ideal" run for each analysis is implemented with the same mesh, the same load increments and the same global solution technique, but the fifth-order Runge-Kutta-England substepping scheme is used to integrate the constitutive law.

The tolerance (i.e. TOL_{sub}) is set to be very small (10^{-10}) to ensure that the number of the substeps is sufficiently high so that the drift from the yield surface can be ignored. These 'ideal' results are then compared with the Gill's fourth-order Runge-Kutta method developed in this research and Modified Euler scheme[8]. Results will be presented for different tolerances. The errors in the elasto-plastic stresses are computed using:

$$Error = \frac{\left[\sum_{i=1}^{N} (\sigma_i - \sigma_i{}^{ideal})^2\right]^{1/2}}{\left[\sum_{i=1}^{N} (\sigma_i{}^{ideal})^2\right]^{1/2}}$$

To test the accuracy and efficiency of the substepping scheme, two analyses for different problem sizes were carried out, one for the problem with 288 d.o.f and 1620 integration points, another for the problem with 432 d.o.f and 3240 integration points. For each analysis, four different models are studied, that is

1. Elastic-perfectly plastic, i.e. no strain hardening model. No stress correction is employed;
2. Elastic-perfectly plastic model with stress correction;
3. Model with the linear strain hardening, but no stress correction;
4. Model with the linear strain hardening, and stress correction is employed.

We use the 8-noded 3D solid element for all analyses. A three-point Gaussian integration rule is considered. The material properties used in the examples are

$$\begin{aligned} E &= 21 \times 10^6 \ psi \\ \nu &= 0.3 \\ H' &= 22 \times 10^4 \ psi \\ \sigma_Y{}^0 &= 24000 \ psi \end{aligned}$$

Note that in all the following figures and tables, '$****$' denotes the diverged solution. ME and RK denote the Modified Euler scheme and Gill's Runge-Kutta scheme, respectively. MEC and RKC mean that for both schemes, stress correction is applied. NH denotes the elastic-perfect plastic model and WH denotes the model which involves strain hardening.

2.5.1 Accuracy

In Figures 2-8 – 2-11, we present force-displacement curve for two analyses for different problem sizes. With reference to Figures 2-8 - 2-11, we can find that, for the cases without strain hardening, the results computed by the schemes without stress correction are almost same as the results from the schemes with stress correction if the tolerance TOL_{sub} is set to be smaller than 10^{-3}. However, for the model in which the strain hardening is considered, the results are completely different if the stress correction is employed. In fact, without stress correction, both schemes can not give the satisfactory accuracy for the cases which involve strain hardening when TOL_{sub} is less than 10^{-5}. For the analysis with 432 d.o.f, the error is so significant that the overall solution diverges. This implies that, due to the approximation nature of the finite element method, yield surface drift may occur with the stresses moving away from the yield surface. This deviation is practically independent of the integration scheme adopted. When the model involves strain (work) hardening where the yield surface is moving with loading increment, the drift is more significant. Since such discrepancies are usually cumulative, it is important to ensure the the stresses are corrected back to the current yield surface at each step of the calculation. Since we apply the stress correction at the end of each substep, it does not affect the accuracy significantly. However, it can make the computed stress fulfil the plasticity criterion at the end of load increment (iteration) and avoid error accumulation and therefore instabilities in the following load steps. This can be seen from the Figures 2-8 – 2-11. For large value of TOL_{sub}, the overall solution diverges due to large error even with the stress correction. The errors in the computed stresses from different algorithms with different tolerances are listed in Tables 2.3 and 2.4.

With reference to the Tables 2.3 and 2.4, it can be seen that, for the case with no strain hardening, we can get the accuracy without stress correction which is adequate for the engineering computation if TOL_{sub} is set to be smaller than 10^{-3}. This accuracy can be improved when the stress correction is employed. However for the analyses with strain hardening, both the Runge-Kutta scheme and the Modified Euler scheme can not give an improved accuracy as TOL_{sub} decreases. This is illustrated in Table 2.3. The problem with 288 d.o.f, the errors are large no matter how TOL_{sub} is reduced. For the problem with 432 d.o.f which involves more integration points, large magnitude

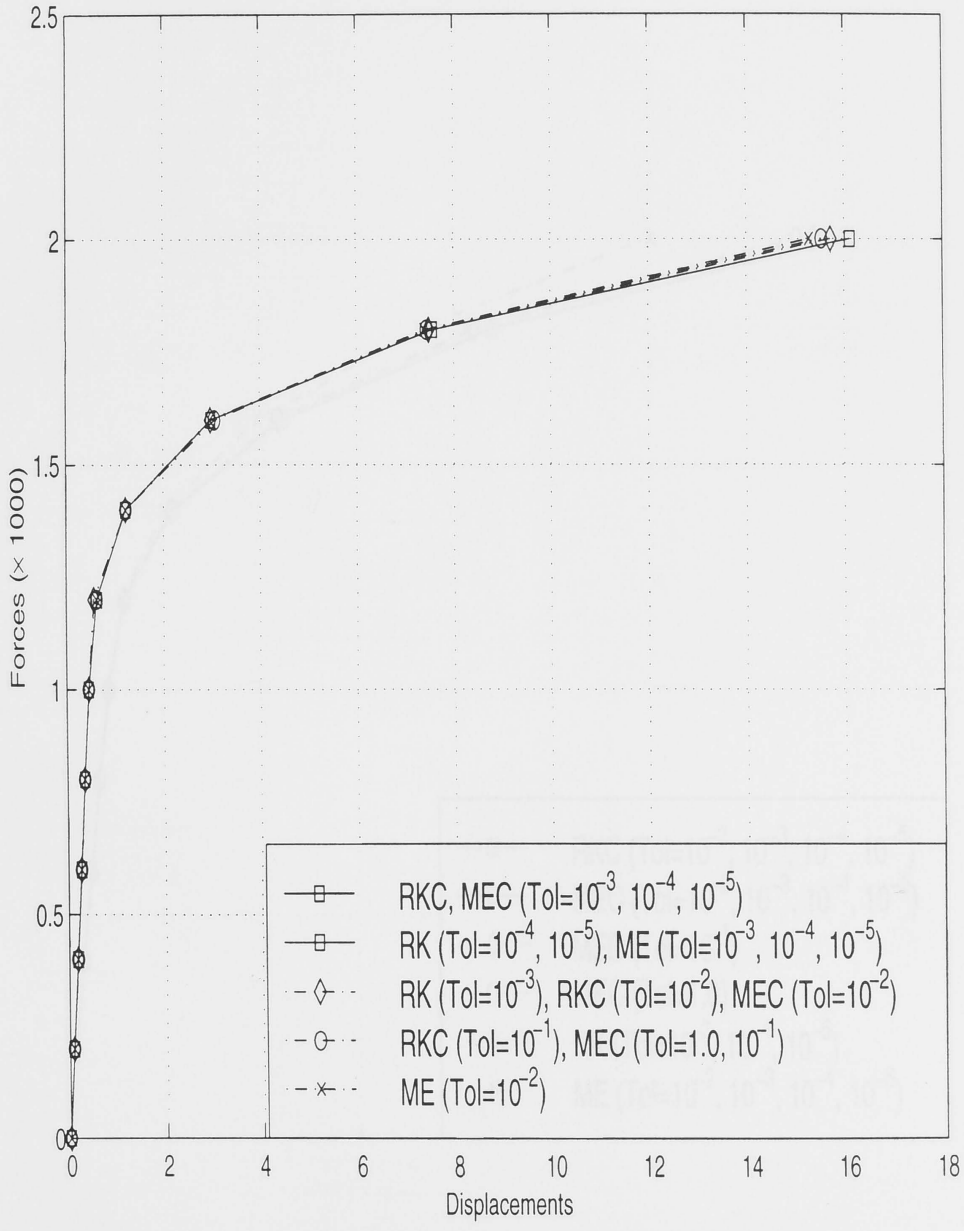

Figure 2-8: Force vs displacement curve for problem with 288 d.o.f (no strain hardening).

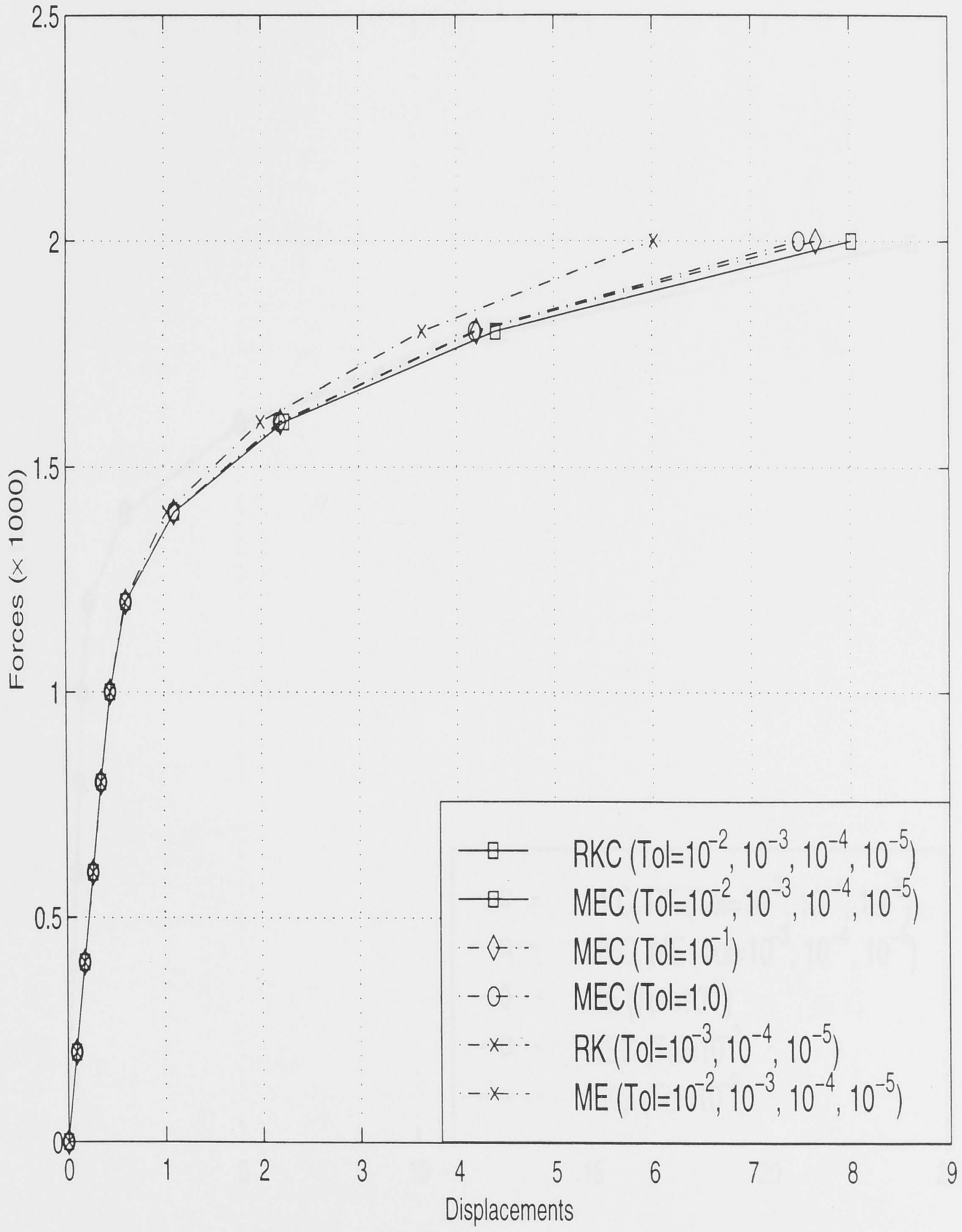

Figure 2-9: Force vs displacement curve for problem with 288 d.o.f (a linear strain hardening is considered).

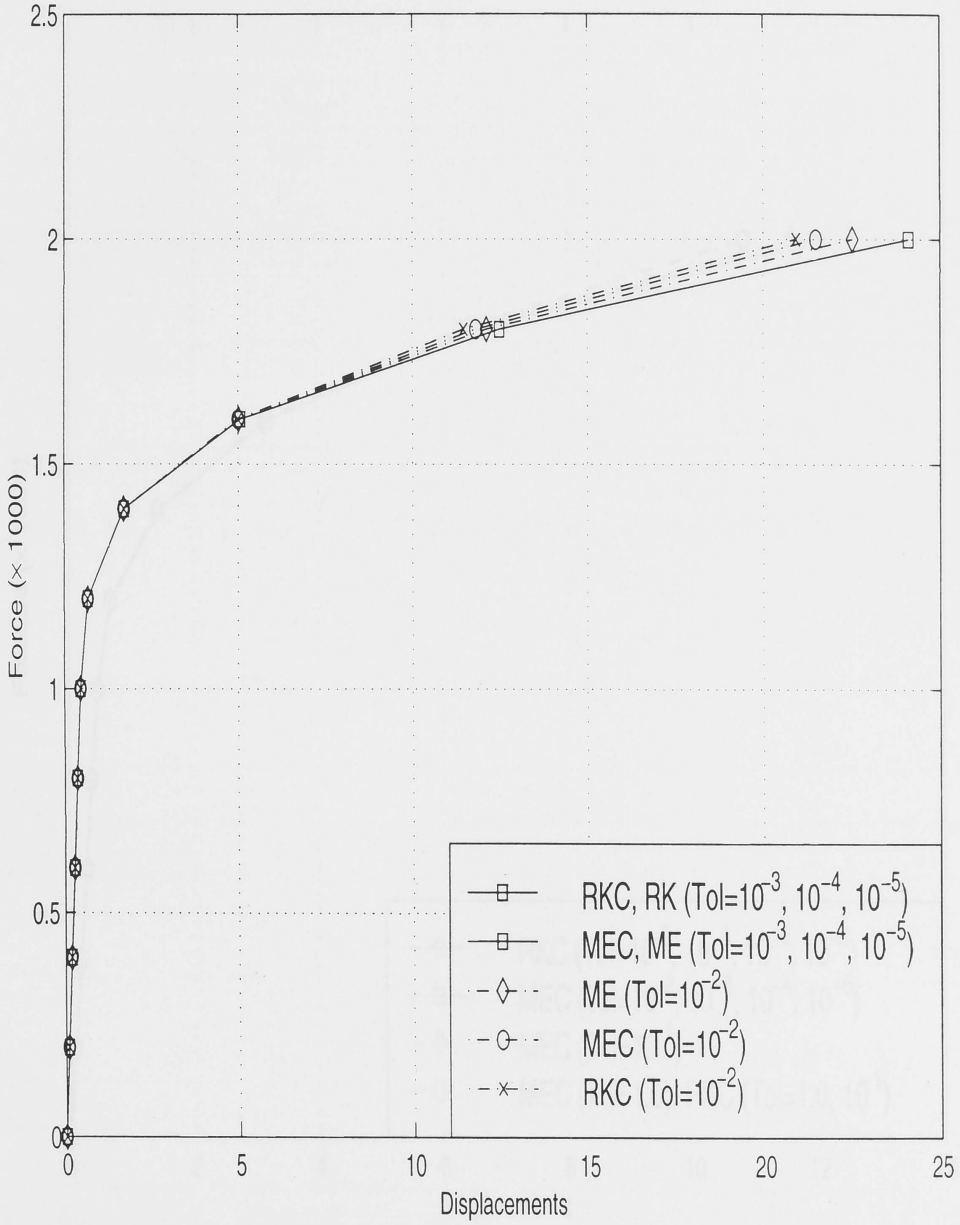

Figure 2-10: Force vs displacement curve for problem with 432 d.o.f (no strain hardening).

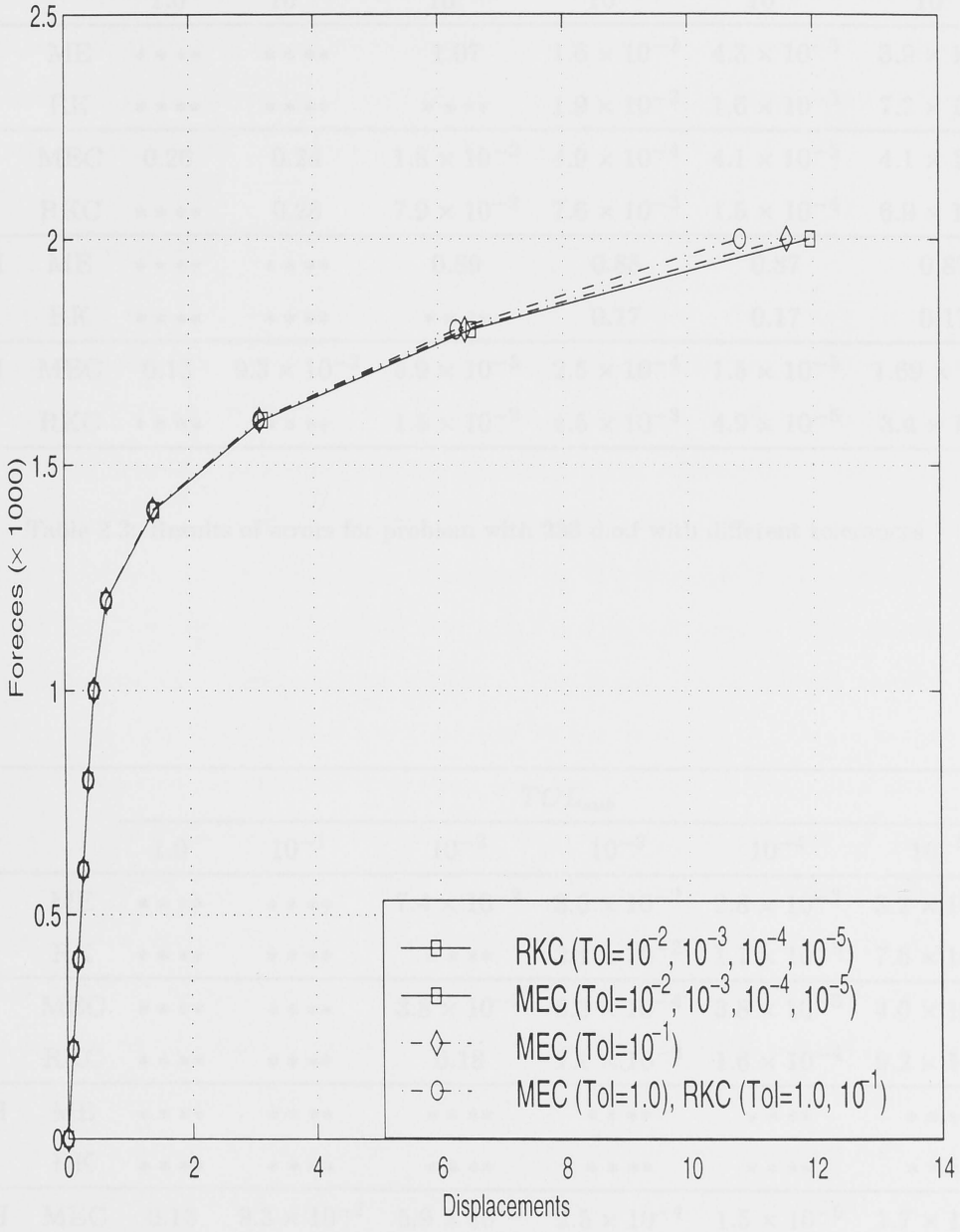

Figure 2-11: Force vs displacement curve for problem with 432 d.o.f (a linear strain hardening is considered).

		TOL_{sub}					
		1.0	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}
NH	ME	****	****	1.07	1.6×10^{-2}	4.3×10^{-3}	3.9×10^{-4}
	RK	****	****	****	1.9×10^{-2}	1.6×10^{-3}	7.2×10^{-4}
NH	MEC	0.26	0.24	1.8×10^{-2}	4.9×10^{-4}	4.1×10^{-5}	4.1×10^{-6}
	RKC	****	0.28	7.9×10^{-2}	7.6×10^{-3}	1.5×10^{-4}	6.9×10^{-6}
WH	ME	****	****	0.89	0.88	0.87	0.87
	RK	****	****	****	0.17	0.17	0.17
WH	MEC	0.13	9.3×10^{-2}	5.9×10^{-3}	2.5×10^{-4}	1.5×10^{-5}	1.69×10^{-6}
	RKC	****	****	1.5×10^{-2}	4.5×10^{-3}	4.9×10^{-5}	3.4×10^{-5}

Table 2.3: Results of errors for problem with 288 d.o.f with different tolerances

		TOL_{sub}					
		1.0	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}
NH	ME	****	****	7.4×10^{-2}	3.6×10^{-2}	2.8×10^{-3}	3.2×10^{-4}
	RK	****	****	****	2.1×10^{-2}	1.1×10^{-3}	7.6×10^{-4}
NH	MEC	****	****	3.8×10^{-2}	6.3×10^{-4}	3.8×10^{-5}	4.0×10^{-6}
	RKC	****	****	0.18	1.1×10^{-2}	1.6×10^{-4}	9.2×10^{-6}
WH	ME	****	****	****	****	****	****
	RK	****	****	****	****	****	****
WH	MEC	0.13	9.3×10^{-2}	5.9×10^{-3}	2.5×10^{-4}	1.5×10^{-5}	1.7×10^{-6}
	RKC	0.19	0.14	2.6×10^{-2}	2.0×10^{-3}	1.8×10^{-4}	7.3×10^{-6}

Table 2.4: Results of errors for problem with 432 d.o.f with different tolerance

of errors cause each analysis to diverge. It is important to note that, when stress correction is used, the accuracy can be improved drastically. One of the conclusions of Sloan's work on the substepping scheme[8] is that stress correction is not required to improve the accuracy. However, the results of the present study clearly indicate the importance of the role of stress correction in the substepping scheme. The accuracy of the results can be improved by at least an order of magnitude by using stress correction for elasto-plastic problem involving strain hardening.

2.5.2 Efficiency

In Tables 2.5 and 2.6, we list the total number of substeps for overall solution of different analysis. The corresponding CPU time spent on the computation of strain-stress relations is listed in Tables 2.7 and 2.8. We can see that, the Runge-Kutta scheme generally requires less substeps than the Modified Euler scheme for a given value of TOL_{sub}. For TOL_{sub} which are equal to 10^{-3} and 10^{-4}, the Runge-Kutta scheme uses less than half of number of substeps consumed by the Modified Euler scheme. This confirms that the high-order Runge-Kutta scheme does not require more substeps to obtain high level of accuracy in the solution. The Runge-Kutta scheme usually uses less CPU time than the Modified Euler scheme for the cases where TOL_{sub} are greater than 10^{-5}. Application of stress correction does not increase the computational time significantly, but it does improve the accuracy considerably. For example, in Table 2.7, for the value of TOL_{sub} equal to 10^{-4}, the CPU time used for the analysis with no stress correction for the Runge-Kutta scheme and the Modified Euler scheme are 1.24 seconds and 1.53 seconds, respectively. However, by using the stress correction, both schemes only use 0.54 and 0.8 second, respectively, to achieve the same level of accuracy. For another example, in Table 2.8, for the application with 432 d.o.f, without stress correction, the Runge-Kutta scheme and the Modified Euler scheme take 3.08 seconds and 3.88 seconds, respectively, to reach an error level of 10^{-4}. However, when the stress correction is employed, the Modified Euler scheme takes 1.48 seconds and the Runge-Kutta scheme takes 1.58 seconds.

		TOL_{sub}					
		1.0	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}
NH	ME	****	****	16006	33790	85734	246458
	RK	****	****	****	13154	34310	131360
NH	MEC	15266	14356	17384	33822	85792	247236
	RKC	****	14506	11156	13394	34310	131460
WH	ME	****	****	7422	15918	42166	123714
	RK	****	****	****	6360	17762	65992
WH	MEC	6258	6390	9498	20872	53640	157198
	RKC	****	****	6090	7782	21670	83616

Table 2.5: Total substeps needed in the overall solution for problem with 288 d.o.f with different tolerance

		TOL_{sub}					
		1.0	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}
NH	ME	****	****	40388	82688	205652	585420
	RK	****	****	****	32748	83384	309676
NH	MEC	****	****	41804	82900	206028	586384
	RKC	****	****	31296	34224	83600	310268
WH	ME	****	****	****	****	****	****
	RK	****	****	****	****	****	****
WH	MEC	14988	18512	23100	49284	128380	372718
	RKC	13968	14424	14724	19536	50824	194368

Table 2.6: Total substeps needed in the overall solution for problem with 432 d.o.f with different tolerance

		TOL_{sub}					
		1.0	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}
NH	ME	****	****	0.50	0.56	0.82	1.24
	RK	****	****	****	0.55	0.79	1.53
NH	MEC	0.86	0.60	0.54	0.54	0.80	1.28
	RKC	****	0.74	0.62	0.52	0.80	1.52
WH	ME	****	****	0.38	0.36	0.52	0.78
	RK	****	****	****	0.31	0.48	0.98
WH	MEC	0.48	0.38	0.42	0.48	0.54	0.90
	RKC	****	****	0.44	0.42	0.49	1.08

Table 2.7: CPU time (seconds)spent on computation of stress-strain relation for problem with 288 d.o.f with different tolerance

		TOL_{sub}					
		1.0	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}
NH	ME	****	****	1.20	1.20	1.68	3.08
	RK	****	****	****	1.20	1.63	3.88
NH	MEC	****	****	1.02	1.48	1.58	3.0
	RKC	****	****	1.38	1.40	1.58	3.9
WH	ME	****	****	****	****	****	****
	RK	****	****	****	****	****	****
WH	MEC	0.90	0.92	0.88	1.02	1.26	2.22
	RKC	0.96	0.96	1.08	1.00	1.25	2.54

Table 2.8: CPU time (seconds) spent on computation of stress-strain relation for problem with 432 d.o.f with different tolerance

2.6 Summary

In this chapter, a modified substepping scheme has been developed and its performance is compared with Sloan's modified Euler scheme. Both substepping schemes control the error in the integration process by permitting the size of each substep to vary in accordance with the behaviour of the constitutive law. The numerical examples indicate that, for the three-dimensional problems for which the strain hardening is considered, the accuracy can not be improved significantly as TOL_{sub} decreases. So for problems in which very high accuracy must be maintained, some forms of stress correction must be employed in order to ensure that the computed stresses remain on the yield surface at any time. Also, applying the stress correction at the end of integration does not increase the computational time significantly, but it can improve the accuracy if it is used with the substepping scheme. In summary, the experience suggests that

1. For elastic-perfectly plastic problems, an error tolerance (TOL_{sub}) in the range of 10^{-4} to 10^{-6} is appropriate for engineering computations. No stress correction is required.
2. For the problems which involve strain (work) hardening, if high accuracy must be maintained, a low value of tolerance should be used to ensure that the stress state does not depart far from the yield surface at any time during the analysis.
3. For both cases, the higher value of tolerance combined with the stress correction is more efficient than choosing small value of tolerance.

In the following parallel implementation of finite element elasto-plastic analysis, the Runge-Kutta scheme will be used to integrate the strain-stress relations.

Chapter 3

Parallel Finite Element Analysis

In this chapter, we will review the history of parallel computing and its role in finite element analysis. The basic formulation of parallel finite element analysis will be presented. Based on the sequential preconditioned conjugate gradient method, we will develop a new parallel algorithm for solving the linearized system of equations. Also, the finite element transformation relationships will be introduced to form a bridge between the sequential algorithm and parallel algorithm.

3.1 Parallel Computing

3.1.1 Introduction

A parallel computer is a set of processors that are able to work cooperatively to solve a computational problem. This definition is broad enough to include parallel supercomputers that have large number of processors, networks of workstations, multipleprocessor workstations, and embedded systems. Parallel computers are interesting because they offer the potential to concentrate computational resources in terms of processors, memory, or I/O bandwidth, on important computational problems.

Parallelism has sometimes been viewed as a rare and exotic subarea of computing, interesting but of little relevance to the average programmer. A study of trends in applications, computer architecture, and networking shows that this view is no longer tenable. Parallelism is becoming ubiquitous, and parallel programming is becoming central to the programming enterprise.

3.1.2 The Importance of Parallel Computing

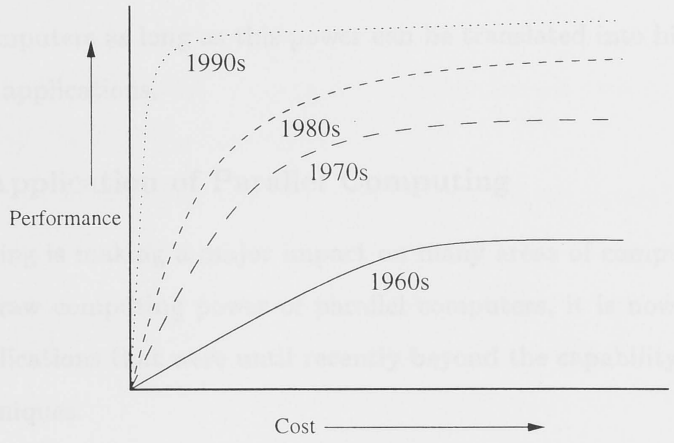

Figure 3-1: Cost versus performance curve and its evolution over the decades

As computers become ever faster, it can be tempting to suppose that they will eventually become "fast enough" and that appetite for increased computing power will be sated. A natural way to circumvent this saturation is to use an ensemble of processors to solve problems. A cost-performance comparison of serial computers over the last few decades shows an interesting evolutionary trend. Figure 3-1[53] represents typical cost-performance curves of serial computers over the past three decades. At the lower end of each curve, performance increases almost linearly (or faster than linearly) with cost. However, beyond a certain point, each curve starts to saturate, and even small gains in performance come at an exorbitant increase in cost. Furthermore, this transition point has become sharper with the passage of time, primarily as a result of advances in very large scale integration (VLSI) technology. It is now possible to construct very fast, low-cost processors. This increases the demand for and production of these processors, resulting in lower prices.

Currently, the speed of off-the-shelf microprocessors is within one order of magnitude of the speed of the fastest serial computers. However, microprocessors cost many orders of magnitude less. This implies that, by connecting only a few microprocessors together to form a parallel computer, it is possible to obtain raw computing power comparable to that of the fastest serial computers. Typically, the cost of such a parallel computer is considerably less.

Furthermore, connecting a large number of processors into a parallel computer

overcomes the saturation point of the computation rates achievable by serial computers. Thus, parallel computers can provide much higher raw computation rates than the fastest serial computers as long as this power can be translated into high computation rates for actual applications.

3.1.3 The Application of Parallel Computing

Parallel processing is making a major impact on many areas of computer application. With the high raw computing power of parallel computers, it is now possible to address many applications that were until recently beyond the capability of conventional computing techniques.

In science and engineering, many production decisions must be made on the basis of crude computational models because the computational hardware to conduct full-scale simulations is not readily available. Many applications, such as weather prediction, metal forming, etc, are always modelled by using a approximate numerical solution, such as finite element method (FEM), which require considerable computational requirement. For example, in studies of crashworthiness, impact and penetration, it is not unusual for an analysis to require 100 hours of CPU time even on current vector supercomputers despite the simplicity of the models being studied. Since processing such modelling requires a nontrivial amount of computation, finding solutions to large instances of these problems is beyond the scope of conventional sequential computing. So it appears to be an inescapable fact that future computers will contain more than one processing unit, and that all processing units will be capable of working in concert on the same problem. This implies that future FEM systems will be required to execute in such parallel environments to adequately support the needs of design optimization and synthesis, nonlinear and crash dynamics and global performance analysis.

3.1.4 Issues in Parallel Computing

To use parallel computing effectively, we need to examine the following issues:

- *Design of parallel computers.* It is important to design parallel computers that can scale up to a large number of processors and are capable of supporting fast communication and data sharing among processors. This is one aspect of parallel computing that has seen the most advances and is the most mature.

- ***Design of efficient algorithms.*** A parallel computer is of little use unless efficient parallel algorithms are available. The issues in designing parallel algorithms are very different from those in designing their sequential counterparts. A significant amount of work is being done to develop efficient parallel algorithms for a variety of parallel architectures.
- ***Methods for evaluating parallel algorithms.*** Given a parallel computer and a parallel algorithm running on it, we need to evaluate the performance of the resulting system. Performance analysis allows us to answer questions such as *How fast can a problem be solved using parallel processing?* and *How efficiently are the processors used?*
- ***Parallel computer languages.*** Parallel algorithms are implemented on parallel computers using a programming language. This language must be flexible enough to allow efficient implementation and must be easy to program in. New languages and programming paradigms are being developed that try to achieve these goals.
- ***Parallel programming tools.*** To facilitate the programming of parallel computers, it is important to develop comprehensive programming environments and tools. These must serve the dual purpose of shielding users from low-level machine characteristics and providing them with design and development tools such as debuggers and simulators.
- ***Automatic programming of parallel computers.*** Much work is being done on the design of parallelizing compilers, which extract implicit parallelism from programs that have not been parallelized explicitly. Such compilers are expected to allow us to program a parallel computer like a serial computer.

3.2 Parallel Finite Element Analysis

The **finite element method** is an active application area of massively parallel computing since the use of finite element methods involves very large amounts of computations. As mentioned before, a typical finite element program consists of a few well-defined modules, namely:

1. generation of element stiffness matrices;
2. assembly and solution of the global system equations;
3. calculations of element characteristics (such as strains, stresses, etc.).

In the elasto-plastic finite element analysis, these modules must interact in an iterative manner. The above stages form three most computationally expensive phases of the FEM. For large-scale three-dimensional non-linear problems, up to 95 percent of total CPU time is consumed in these three phases.

3.2.1 Generation of Element Stiffness Matrices

The computation of an elemental stiffness matrix requires the numerical integration, which is performed by Gaussian quadrature. On a sequential computing system, the finite element code for evaluating the elemental stiffness matrices for all elements in the mesh has the following structure:

```
loop over all finite elements
    loop over all quadrature points
        evaluate Jacobian and shape function
          derivatives for current quadrature point
        loop over rows in elemental stiffness matrix
            loop over columns in elemental stiffness matrix
                evaluate contribution to entry (row, column)
                  of the elemental stiffness matrix
            end loop
        end loop
    end loop
end loop
```

3.2.2 Assembly and Solution of the Global System Equations

Techniques to solve the equations system may generally be classified into two categories, one is the direct solution and another is the iterative solution. Iterative technique requires less storage than direct solver for most three-dimensional applications. With an improved understanding of preconditioned techniques for the conjugate gradient method, the computational efficiency of iterative techniques makes them very competitive with direct solver. Briefly, the computations of the preconditioned conjugate gradient method are:

Assemble the global matrix and vector
Initialize displacement vector
repeat until convergence
    compute residual vector
    apply boundary conditions
    compute acceleration parameters
    evaluate new estimate of displacement vector
end loop

For large-scale three-dimensional analysis, the resulting system of linear equations is large and sparse, and hence solving it is the most computationally expensive phase of the FEM. It is this phase for which efficient parallel solutions are critical.

3.2.3 Calculations of Element Characteristics

After the displacements are determined, the strains and stresses can be computed. The procedure is listed as following:

loop over all finite elements
    loop over all quadrature points
        evaluate Jacobian and shape function
            derivatives for current quadrature point
        determine number of integration substeps
            for current quadrature point
        loop over all integration substeps
            evaluate stress for current quadrature point
        end loop
        loop over all degree of freedoms in an element
            evaluate equivalent nodal forces for current element
        end loop
    end loop
end loop

It can be seen from the procedures listed above that most the computations involved in a finite element analysis are carried out at the element level. These include the formation of element stiffnesses and the computation of elemental derivatives. If the

whole domain is divided in such a way that each processor represents a certain number of elements, consequently, these operations can be carried out in parallel without any synchronization. However, after the elemental calculations have been achieved, the calculation of the displacements can become a bottleneck for parallel implementation, since the global matrix and vector need to be assembled at this stage. It would be very natural, especially from a parallel processing view point, if the selected numerical solution could also operate at the element (substructure) level. Generally, this is the case for iterative solution schemes based on matrix-vector products. It is well known that such products can be achieved without the need for assembling the global matrix and vector[49]. The most well-known iterative method, i.e. Preconditioning Conjugate Gradient (PCG) method is used in this research work.

3.3 Sequential Algorithm for Equation Solution

The principal computational effort in a finite element analysis is to solve the linear system of equations of the form

$$[K]\{x\} = \{f\} \tag{3.1}$$

in which $[K]$, $\{x\}$ and $\{f\}$ are the global stiffness matrix, nodal displacement vector, and nodal force vector, respectively.

Generally, techniques to solve this equations system may generally be classified into direct and iterative methods. Direct methods can be applied to any non-singular matrix, and are well adapted to matrix inversion and solution of linear equations with many right hand sides. These methods are especially well suited to solving dense systems. Their usefulness for general sparse systems is limited by the following fact: more storage is wasted on storing the zero fill-ins which cause additional computational cost. For very large, sparse matrices this can lead to a significant penalty in storage. In contrast, the use of iterative methods for the solution of a linearized system of equations has two desirable advantages: (1) it efficiently exploits the sparsity of the involved matrices and therefore requires less storage than direct schemes; (2) it provides a means of controlling the accuracy of the solution. By avoiding the operations associated with zero fill-ins, not only the storage requirements but also the computational cost can be reduced. So the iterative methods are particularly attractive for solving the matrix

with large sparsity. The Preconditioned Conjugate Gradient (PCG) has emerged over the last decades as a favourite algorithm for solving large sparse system of equations.

3.3.1 Conjugate gradient method

The Conjugate Gradient (CG) method was proposed in 1952 for the solution of linear systems with a symmetric, positive definite matrix. The method can be considered as direct method because the exact solution is obtained at most in the step n in the absence of round-off errors. The CG method belongs to a class of iterative methods known as ***minimization methods***. An iteration of a minimization is normally of the form

$$\{x\}_k = \{x\}_{k-1} + \alpha_k \{p\}_k \tag{3.2}$$

where α_k is a scalar step size and $\{p\}_k$ is the direction vector. For a given x_{k-1} and p_k, the scalar α_k is chosen to mimimize the norm of the residual vector $\{r\}$ where

$$\{r\}_k = \{f\} - [K]\{x\}_k \tag{3.3}$$

and it can be determined by

$$\alpha_k = \frac{\{p\}_k^T \{r\}_{k-1}}{\{p\}_k^T [K] \{p\}_k} \tag{3.4}$$

The residual vector need not be computed explicitly in each iteration because it can be computed incrementally by using its value from the previous iteration, that is,

$$\begin{aligned} \{r\}_k &= \{f\} - [K]\{x\}_k \\ &= \{f\} - [K](\{x\}_{k-1} + \alpha_k \{p\}_k) \\ &= \{f\} - [K]\{x\}_{k-1} - \alpha_k [K]\{p\}_k \\ &= \{r\}_{k-1} - \alpha_k [K]\{p\}_k \end{aligned} \tag{3.5}$$

Thus, the only matrix-vector product computed in each iteration is $[K]\{p\}_k$, which is already required to compute α_k (Equation 3.4).

If $[K]$ is a symmetric positive definite matrix and $p_1, p_2, \cdots, p_n$ are direction vectors that are conjugate with respect to $[K]$, then $\{x\}$ converge to the solution of $[K]\{x\} = \{f\}$ in at most n iterations, assuming no rounding errors. In practice, however, the number of iterations that yields an acceptable approximation to the solution is much

smaller than n. At the beginning of the CG algorithm, the set of vectors is chosen as follows:

$$\{x\}_0 = 0, \quad \{r\}_0 = \{p\}_0 = \{f\}$$

and the direction vectors are chosen according to

$$\{p\}_{k+1} = \{r\}_k + \frac{\{r\}_k^T \{r\}_k}{\{r\}_{k-1}^T \{r\}_{k-1}} \{p\}_k \tag{3.6}$$

The algorithm terminates when the two-norm of the current residual falls below a predetermined fraction of the two-norm of the initial residual $\{r\}_0$.

3.3.2 Preconditioned conjugate gradient method

The speed of convergence of the CG algorithm can be increased through the introduction of the technique called preconditioning. The CG method can then be applied to a preconditioned system:

$$[C]^{-1}([K]\{x\} - \{f\}) = \{0\} \tag{3.7}$$

where $[C]$ is referred to as the conditioning matrix. The reason is that the rate of convergence of the Equation 3.7 will, in general, be higher than that of Equation 3.1, because the spectrum of the coefficient matrix will be included in a disk with smaller radius than that of the original matrix, and the smaller the radius, the higher the rate of convergence. The resulting algorithm is always referred as to Preconditioned Conjugate Gradient (PCG) method. There exist various choices of the conditioning matrix $[C]$, from the diagonal entries of matrix $[K]$ to some forms of incomplete Cholesky factor of matrix $[K]$. Selection of an optimal preconditioner remains a topic of much current research.

It should be noted that unless $[C]$ is a diagonal matrix, the sparsity pattern of $[K]$ is not preserved. Since a diagonal storage scheme will be used to exploit the sparsity of $[K]$, a diagonal preconditioner will be used in this thesis.

3.3.3 Minimal Residual Smoothing

It is known from practice that the norm of the residuals of the CG method may heavily oscillate. Therefore, the smoothing algorithm was introduced in order to get a

non-increasing norm of the residuals. In each iteration the minimization is performed following the CG method according to

$$\begin{aligned} \{s\}_0 &= \{r\}_0 \\ \{z\}_0 &= \{x\}_0 \\ \{s\}_k &= \{s\}_{k-1} + \delta_k(\{r\}_k - \{s\}_{k-1}) \qquad (3.8) \\ \{z\}_k &= \{z\}_{k-1} + \delta_k(\{x\}_k - \{z\}_{k-1}) \qquad (3.9) \end{aligned}$$

where $\{x\}_k$ and $\{r\}_k$ are the solution and the residual for k^{th} iteration, respectively, resulting from the CG method. The different smoothing techniques vary by the determination of the coefficient δ_k. In this thesis, a technique which is called Minimal Residual (MR) smoothing method is employed. The principle of this method is to minimize the norm of residuals according to

$$\| \{s\}_k \|_2 = min \| \{s\}_{k-1} + \delta_k(\{r\}_k - \{s\}_{k-1}) \|_2 \qquad (3.10)$$

So we get

$$\delta_k = \frac{\{s\}_{k-1}^T(\{r\}_k - \{s\}_{k-1})}{(\{r\}_k - \{s\}_{k-1})^T(\{r\}_k - \{s\}_{k-1})} \qquad (3.11)$$

A direct result of Equation 3.10 is that the smoothed residuals are a function of the iteration index with a non-increasing norm, i. e.

$$\begin{aligned} \| \{s\}_k \|_2 &\leq \| \{s\}_{k-1} \|_2 \qquad (3.12) \\ \| \{s\}_k \|_2 &\leq \| \{r\}_k \|_2 \qquad (3.13) \end{aligned}$$

When the MR smoothing is used, $\{z\}$ is used for the following computation and the stopping criterion becomes

$$\frac{\| \{s\}_k \|_2}{\| \{r\}_0 \|_2} < Tolerance \qquad (3.14)$$

Introducing temporary vectors

$$\{h\} = [K]\{p\}, \quad \{t\} = [C]^{-1}\{r\}$$

the sequential version of the algorithms, including PCG and MR smoothing, can be summarized in Table 3.1.

Table 3.1: PCG algorithm with MR smoothing

Initialization:

$$
\begin{aligned}
\{x\}_0 &:= \{0\} \\
\{z\}_0 &:= \{0\} \\
\{r\}_0 &:= \{f\} - [K]\{x\}_0 \\
\{t\}_0 &:= [C]^{-1}\{r\}_0 \\
\{s\}_0 &:= \{t\}_0 \\
\{p\}_0 &:= \{t\}_0
\end{aligned}
$$

Iterate $k = 1,2,\cdots$ If $\|\{s\}_k\|/\|\{f\}\| < tolerance$ terminate

$$
\left.
\begin{aligned}
\{h\}_k &:= [K]\{p\}_{k-1} \\
\rho_{k-1} &:= \{t\}_{k-1}^T\{r\}_{k-1} \\
\beta_k &:= \{p\}_{k-1}^T\{h\}_k \\
\alpha_k &:= \rho_{k-1}/\beta_k \\
\{x\}_k &:= \{x\}_{k-1} + \alpha_k\{p\}_{k-1} \\
\{r\}_k &:= \{r\}_{k-1} - \alpha_k\{h\}_k \\
\{t\}_k &:= [C]^{-1}\{r\}_k \\
\rho_k &:= \{t\}_k^T\{r\}_k \\
\{p\}_k &:= \{t\}_k + \tfrac{\rho_k}{\rho_{k-1}}\{p\}_{k-1}
\end{aligned}
\right\} PCG
$$

$$
\left.
\begin{aligned}
\mu_k &:= \{s\}_{k-1}^T(\{t\}_k - \{s\}_{k-1}) \\
\theta_k &:= (\{t\}_k - \{s\}_{k-1})^T(\{t\}_k - \{s\}_{k-1}) \\
\delta_k &:= -\mu_k/\theta_k \\
\{s\}_k &:= \{s\}_{k-1} + \delta_k(\{t\}_k - \{s\}_{k-1}) \\
\{z\}_k &:= \{z\}_{k-1} + \delta_k(\{x\}_k - \{z\}_{k-1})
\end{aligned}
\right\} MR \quad SMOOTHING
$$

3.4 Finite Element Transformation Relations

The displacement method of structural or finite element analysis can be derived from the characteristic, compatibility and equilibrium relations. The characteristic relation can be stated as:

$$\left\{f^{(e)}\right\} = \left[K^{(e)}\right]\left\{x^{(e)}\right\} \tag{3.15}$$

where $\left[K^{(e)}\right]$, $\left\{f^{(e)}\right\}$ and $\left\{x^{(e)}\right\}$ are, respectively, the element stiffness matrix, element

force vector and element distortion vector for a finite element e. The global displacement vector $\{x\}$ and the element distortion vector $\{x^e\}$ are related as:

$$\{x^e\} = [A]\{x\} \tag{3.16}$$

where $\{x^e\} = \lfloor \lfloor x^{(1)} \rfloor, \cdots, \lfloor x^{(e)} \rfloor, \cdots, \lfloor x^{(p)} \rfloor \rfloor^t$ and $[A]$ is termed the global kinematics matrix. The global load vector $\{f\}$ and the element force vector $\{f^e\}$ are related as

$$\{f\} = [A]^t \{f^e\} \tag{3.17}$$

where $\{f^e\} = \lfloor \lfloor f^{(1)} \rfloor, \cdots, \lfloor f^{(e)} \rfloor, \cdots, \lfloor f^{(p)} \rfloor \rfloor^t$ and $[A]^t$ is termed the global statics matrix. Combining Equation 3.15 - 3.17, the system equations can be written as:

$$\{f\} = [A]^t \lceil K^e \rfloor [A]\{x\} = [K]\{x\} \tag{3.18}$$

where $[K]$ is the system stiffness matrix and, $\lceil K^e \rfloor$ is the pseudo-diagonal block matrix consisting of the element stiffnesses $\left[K^{(e)}\right]$.

If the element stiffnesses are oriented in the same co-ordinate as the global system, the kinematics matrix $[A]$ is a Boolean matrix populated with zeros and unit values. Specifically, in column j of $[A]$, a unit value appears at the row locations corresponding to the local nodes of the finite elements incident at node j of the global system. Furthermore, it can easily be shown that

$$[A][A]^t = \lceil I \rfloor + [X] \tag{3.19}$$

where $[X]$ is a symmetric Boolean matrix. In each row of $[X]$, unit values appear in the column locations corresponding to the incident nodes of the neighbouring finite elements. The matrix product $[A][A]^t$ represents the element-element connectivity information. This result is illustrated in Figure 3-2.

Let $\{q\}$ and $\{q^e\} = \lfloor \lfloor q^{(1)} \rfloor, \cdots, \lfloor q^{(e)} \rfloor, \cdots, \lfloor q^{(p)} \rfloor \rfloor^t$ represent a global vector and an element vector, respectively. The operation $[A]^t\{q^e\}$ is equivalent to summing the local quantities from all incident elements to the global system. Similarly, the operation $[A]\{q\}$ denotes distributing the quantity from the global system to the local elements. Hence, the quantity $\{s^e\} = [A][A]^t\{q^e\}$ represents the element quantity $\{q^e\}$ plus those contributed from the neighbouring elements. An element vector $\left\{s^{(e)}\right\}$ can be calculated as:

$$\left\{s^{(e)}\right\} = \sum_{i \in \{adj(e)\}} \left\{q^{(i)}\right\} + \left\{q^{(e)}\right\} \tag{3.20}$$

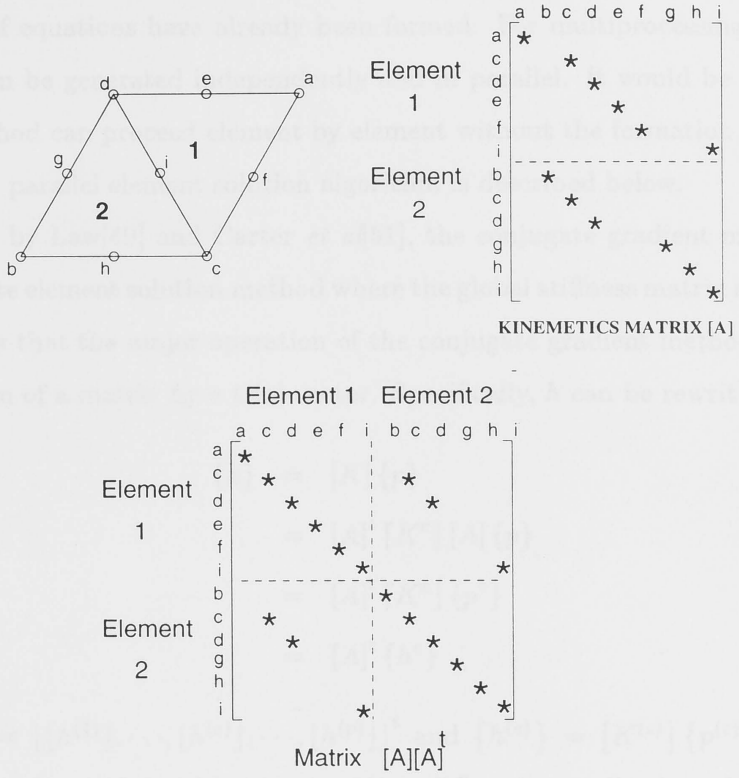

Figure 3-2: Element-element connectivity information.

where $\{adj(e)\}$ represents the set of neighbouring elements adjacent to finite element e.

In an parallel computing system, if we assume each finite element is mapped to a processor, Equation 3.20 is equivalent to the following steps:

(a) send $\{q^{(e)}\}$ to and receive $\{q^{(i)}\}$ from processors storing the neighbouring elements;

(b) $\{s^{(e)}\} = \sum \{q^{(i)}\} \{q^{(e)}\}$

That is, the vector $\{s^e\}$ can be computed by an interprocessor communication and a local sum.

3.5 Parallel Algorithms for Equation Solution

3.5.1 Parallel Preconditioned Conjugate Gradient Method

In selecting a solution scheme from a multiprocessing system, iterative solution methods are often preferred. Most studies of finite element solution algorithms often assume that

the system of equations have already been formed. For multiprocessing, the element stiffnesses can be generated independently and in parallel. It would be natural if the solution method can proceed element by element without the formation of the system equations. A parallel element solution algorithm is described below.

As noted by Law[49] and Carter *et al*[51], the conjugate gradient method can be used as a finite element solution method where the global stiffness matrix need not exist. The reason is that the major operation of the conjugate gradient method involves the multiplication of a matrix by a trial vector. Specifically, h can be rewritten as

$$\begin{aligned} \{h\} &= [K]\{p\} \\ &= [A]^t \lceil K^e \rfloor [A]\{p\} \\ &= [A]^t \lceil K^e \rfloor \{p^e\} \\ &= [A]^t \{h^e\} \end{aligned} \tag{3.21}$$

where $\{h^e\} = \lfloor \lfloor h^{(1)} \rfloor, \cdots, \lfloor h^{(e)} \rfloor, \cdots, \lfloor h^{(p)} \rfloor \rfloor^t$ and $\left\{h^{(e)}\right\} = \left[K^{(e)}\right]\left\{p^{(e)}\right\}$. This multiplication can be performed in steps, adding different contributions from individual elements to the entry of the resulting vector. The matrix-vector multiplication is performed directly on the element level. The resulting element vectors $\left\{h^{(e)}\right\}$ are summed to form the global vector $\{h\}$. This method provides a better control of numerical accuracy, in that cancellation of digits due to the assembling process can be minimized. The conjugate gradient algorithm can be expressed entirely in terms of the element matrix and element vectors. The relation between global vectors and element vectors can be described as follows:

$$\{p^e\} = [A]\{P\} \tag{3.22}$$

$$\{x^e\} = [A]\{x\} \tag{3.23}$$

$$[A]^t\{f^e\} = \{f\} \tag{3.24}$$

and

$$[A]^t\{r^e\} = \{r\} \tag{3.25}$$

From Equation 3.20 and 3.25, the inner product $\{r\}^t\{r\}$ can be rewritten as

$$\begin{aligned}\gamma &= \{r\}^t\{r\} \\ &= \{r^e\}[A][A]^t\{r^e\} \\ &= \{r^e\}^t\{s^e\} \\ &= \sum_{e=1}^{p}\left\{r^{(e)}\right\}^t\left\{s^{(e)}\right\} \\ &= \sum_{e=1}^{p}\rho^{(e)}\end{aligned} \tag{3.26}$$

where

$$\{s^e\} = \lfloor\lfloor s^{(1)}\rfloor, \cdots, \lfloor s^{(e)}\rfloor, \cdots, \lfloor s^{(p)}\rfloor\rfloor^t \tag{3.27}$$

$$\left\{s^{(e)}\right\} = \sum_{i\in\{adj(e)\}}\left\{r^{(i)}\right\} + \left\{r^{(e)}\right\} \tag{3.28}$$

and

$$\rho^{(e)} = \left\{r^{(e)}\right\}^t\left\{s^{(e)}\right\} \tag{3.29}$$

Similarly, the inner product $\{p\}^t\{u\}$ (or $\{p\}^t[K]\{p\}$) can be written as

$$\begin{aligned}\beta &= \{p\}^t[K]\{p\} \\ &= \{p\}^t[A]^t\lceil K^e\rfloor[A]\{p\} \\ &= \{p^e\}^t\{u^e\} \\ &= \sum_{e=1}^{p}\beta^{(e)}\end{aligned} \tag{3.30}$$

where

$$\beta^{(e)} = \left\{p^{(e)}\right\}^t\left\{u^{(e)}\right\} \tag{3.31}$$

Hence, the coefficient α in the sequential algorithm can be computed as

$$\frac{1}{\alpha} = \frac{\sum_{e=1}^{p}\beta^{(e)}}{\gamma} = \sum_{e=1}^{p}\sigma^{(e)} \tag{3.32}$$

where

$$\sigma^{(e)} = \beta^{(e)}/\gamma \tag{3.33}$$

The global displacement vector $\{x\}$ can be distributed to element distortions by

$$[A]\{x_{new}\} = [A]\{x\} + \alpha [A]\{p\} \tag{3.34}$$

or

$$\{x^e_{new}\} = \{x^e\} + \alpha \{p^e\} \tag{3.35}$$

Similarly, the residual vector $\{r\}$ can be rewritten as

$$\begin{aligned} \{r_{new}\} &= [A]^t \{r^e_{new}\} \\ &= [A]^t \{r^e\} - \alpha [A]^t \{u^e\} \\ &= [A]^t (\{r^e\} - \alpha \{u^e\}) \end{aligned} \tag{3.36}$$

Thus, we can denote the element residual vector as

$$\{r^e_{new}\} = \{r^e\} - \alpha \{h^e\} \tag{3.37}$$

Finally, the element conjugate direction $\{p^e_{new}\}$ can be written as

$$\begin{aligned} \{p^e_{new}\} &= [A]\{p_{new}\} \\ &= [A]\{r_{new}\} + \lambda [A]\{p\} \\ &= \{s^e_{new}\} + \lambda \{p^e\} \end{aligned} \tag{3.38}$$

Here, the calculation of the gradient direction vector enforces the compatibility condition at the element level. The residual and the initial solution vectors can be initialized as

$$\{x^e_0\} = 0 \tag{3.39}$$

and

$$\{r^e\} = \{f^e\} - \lceil K^e \rfloor \{x^e_0\} = \{f^e\} \tag{3.40}$$

Using Equations 3.21-3.40, the preconditioned conjugate gradient algorithm can be expressed and computed entirely on an element by element basis.

3.5.2 Parallel Minimal Residual Smoothing method

It can be seen that the computation of scalar δ is important in the MR smoothing algorithm. When computing the scalar δ, two products need to be performed to calculate μ and θ. Similarly, the technique used in the inner product in the parallel PCG algorithm can be applied to parallel MR smoothing algorithm. Let $\{u\}$ denotes $\{t\}-\{s\}$, so that,

$$\begin{aligned}\mu &= \{s\}^T(\{t\}-\{s\}) \\ &= \{s\}^T\{u\} \\ &= \{s^e\}\,[A]\,[A]^t\,(\{u^e\} \\ &= \{s^e\}^t\{v^e\} \\ &= \sum_{e=1}^{p}\{s^{(e)}\}^t\{v^{(e)}\}\end{aligned} \tag{3.41}$$

where

$$\{v^e\} = \lfloor\lfloor v^{(1)}\rfloor,\cdots,\lfloor v^{(e)}\rfloor,\cdots,\lfloor v^{(p)}\rfloor\rfloor^t \tag{3.42}$$

$$\left\{v^{(e)}\right\} = \sum_{i\in\{adj(e)\}}\left\{u^{(i)}\right\} + \left\{u^{(e)}\right\} \tag{3.43}$$

and

$$\{u^{(e)}\} = \{t^{(e)}\}^t - \{s^{(e)}\} \tag{3.44}$$

In a similar manner, θ can be rewritten as

$$\begin{aligned}\mu &= \{u\}^T(\{t\}-\{s\}) \\ &= \{u\}^T\{u\} \\ &= \{u^e\}\,[A]\,[A]^t\,(\{u^e\} \\ &= \{u^e\}^t\{v^e\} \\ &= \sum_{e=1}^{p}\{u^{(e)}\}^t\{v^{(e)}\}\end{aligned} \tag{3.45}$$

where $\{v^{(e)}\}$ and $\{v^{(e)}\}$ are defined as above. After δ is determined, all other computations can be implemented at the element level.

It can be seen that parallel element-by-element algorithm developed above can be easily modified to the substructure-by-substructure algorithm by using finite element

transformation relations. The combination of parallel conjugate gradient algorithm and parallel MR smoothing algorithm forms a new parallel algorithm for the solution of the linearized system of equations. Throughout the process, the formation of global system is not performed. The displacements for each substructure are computed separately by each processor. The storage space required for each processor includes an substructure matrix and vectors. A Parallel substructure Preconditioned Conjugate Gradient algorithm with parallel MR smoothing (PPCGMR) is described in Table 3.2.

Table 3.2: Parallel Substructure Preconditioned Conjugate Gradient Algorithm combined with MR Smoothing

Initialization:

1: (a) $\{x^{(s)}\}_0 = 0$
(b) $\{z^{(s)}\}_0 = 0$
(c) $\{r^{(s)}\}_0 = \{f^{(s)}\}$
(d) Compute $[C]^{(s)}$

2: Exchange $[C^{(s)}]$ with neighbour j

3: (a) $[C^{a(s)}] = \sum_{j \in adj(s)} [C^{(j)}] + [C^{(s)}]$
(b) $\{t^{(s)}\}_0 = [C^{a(s)}]^{(-1)} \{r^{(s)}\}_0$
(c) $\{s^{(s)}\}_0 = \{t^{(s)}\}_0$

4: Exchange $\{t^{(s)}\}_0$ with neighbour j

5: (a) $\{t^{a(s)}\}_0 = \sum_{j \in adj(s)} \{t^{(j)}\}_0 + \{t^{(s)}\}_0$
(b) $\{s^{a(s)}\}_0 = \{t^{a(s)}\}_0$
(c) $\{p^{(s)}\}_0 = \{t^{a(s)}\}_0$
(d) ${\rho_0}^{(s)} = \{t^{a(s)}\}_0^T \{t^{(s)}\}_0$

6: (Merge Sum) $\gamma_0 = \rho_0 = \sum {\rho^{(s)}}_0, \quad s = 1, ..., p$

Iterate $k = 1, 2, \quad \cdots$ If $\gamma_k / \gamma_0 <$ tolerance terminate

1: (a) $\{h^{(s)}\}_k = [K^{(s)}] \{p^{(s)}\}_{k-1}$
(b) ${\beta_k}^{(s)} = \{p^{(s)}\}^T{}_{k-1} \{h^{(s)}\}_k$
(c) ${\sigma_k}^{(s)} = {\beta_k}^{(s)} / \gamma_{k-1}$

2: (Merge Sum) $1/\alpha_k = \sum {\sigma_k}^{(s)}, s = 1, ..., p$

3: (a) $\{x^{(s)}\}_k = \{x^{(s)}\}_{k-1} + \alpha_k \{p^{(s)}\}_{k-1}$
(b) $\{r^{(s)}\}_k = \{r^{(s)}\}_{k-1} - \alpha_k \{h^{(s)}\}_k$

(c) $\{t^{(s)}\}_k = [C^{a(s)}]^{(-1)}\{r^{(s)}\}_k$

4: Exchange $\{t^{(s)}\}_k$ with neighbour j

5: (a) $\{t^{a(s)}\}_k = \sum_{j \in adj(s)} \{t^{(j)}\} + \{t^{(s)}\}$
(b) $\rho_k{}^{(s)} = \{t^{a(s)}\}^T{}_k \{t^{(s)}\}_k$

6: (Merge Sum) $\rho_k = \sum \rho_k{}^{(s)}, s = 1, ..., p$

7: (a) $\{p^{(s)}\}_k = \{t^{a(s)}\}_k + (\rho_k/\rho_{k-1})\{p^{(s)}\}_{k-1}$
(b) $\mu_k{}^{(s)} = \{s^{a(s)}\}^T{}_k (\{t^{(s)}\}_k - \{s^{(s)}\}_k)$
(c) $\theta_k{}^{(s)} = (\{q^{(s)}\}_k - \{g^{(s)}\}_{k-1})^T (\{t^{(s)}\}_k - \{s^{(s)}\}_{k-1})$

8: (a) (Merge Sum) $\mu_k = \sum \mu_k{}^{(s)}$
(b) (Merge Sum) $\theta_k = \sum \theta_k{}^{(s)}$
(c) $\delta_k = -\mu_k/\theta_k$

9: (a) $\{s^{(s)}\}_k = \{s^{(s)}\}_{k-1} + \delta_k(\{t^{(s)}\}_k - \{s^{(s)}\}_{k-1})$
(b) $\{g^{(s)}\}_k = \{s_g^{(s)}\}_{k-1} + \delta_k(\{q^{(s)}\}_k - \{g^{(s)}\}_{k-1})$
(c) $\{z^{(s)}\}_k = \{z^{(s)}\}_{k-1} + \delta_k(\{x^{(s)}\}_k - \{z^{(s)}\}_{k-1})$
(d) $\gamma_k{}^{(s)} = \{s^{a(s)}\}^t{}_k \{s^{(s)}\}_k$

10: (Merge Sum) $\gamma_k = \sum \gamma_k{}^{(s)}$

11: Go To Step 1

*Superscript a denotes the assembled vector.

Chapter 4

Implementation of Parallel Algorithms

In this chapter, we will describe the parallel environment used and many fundamental metrics for evaluating the performance of parallel algorithm which will serve as the basis for subsequent discussion of performance analysis, and implementation. The primary issues for parallel implementation of the finite element method will also be presented. Based on these thoughts, several techniques are used with an aim to get optimal performance.

4.1 Parallel Environment

4.1.1 Parallel Computer Architectures

Parallel Computers

The basic motivation of the new advanced computer systems is to overcome the limitations of Single Instruction-Single Data (SISD) computers, which execute instructions sequentially, by developing new computer architectures suitable for parallel computing. Four broad classifications of the parallel computer architectures can be identified according to their machine organization:

1. Single instruction-multiple data (SIMD) machines. An SIMD machine generally consists of a collection of identical processors, a memory or memories and an

interconnection scheme which allows processors to exchange data. During execution of a program, each processor performs the same instruction sequence, but uses different data.

2. Multiple instruction-single data (MISD) machines. In an MISD machine, the basic arithmetic operations are broken up into a set of elementary steps which, when performed in series, achieve the desired operation. Each elementary step is then implemented into hardware and the resulting arithmetic unit operates as a production line, called the "pipeline".

3. Multiple instruction-multiple data (MIMD) machines. An MIMD machine typically contains a number of interconnected processors, each of which is programmable and can execute its own instructions. The instructions for each processor can be the same or different. The processors operate on a shared memory (or memories), generally in an asynchronous manner.

4. Special purpose systolic machines. Systolic machines are devices attached to a conventional computer to perform special purpose functions with extremely high speed. The basic principle of systolic machines is to replace a single processing element by a network of interconnected processing elements. Each processor regularly pumps data in and out and performs some simple and short computations. A data item, once brought out from memory, is to be used effectively at each processor while passing through the network.

Among all kinds of the parallel computers, the most economic architecture is the workstation cluster. The workstation cluster is physically obtained by linking the single workstation to each other via Ethernet. It has been used as computing resources for quite some time and has become a mature concept with the availability of a large amount of software. They are also finally becoming competitive in performance with proprietary supercomputers.

Linux-Alpha Workstation Cluster

The parallel computer used in the present study is a Linux-Alpha Beowulf Cluster (Figure 4-1). The Linux-Alpha Cluster consists of twelve 533MHz Alpha LX164s each with

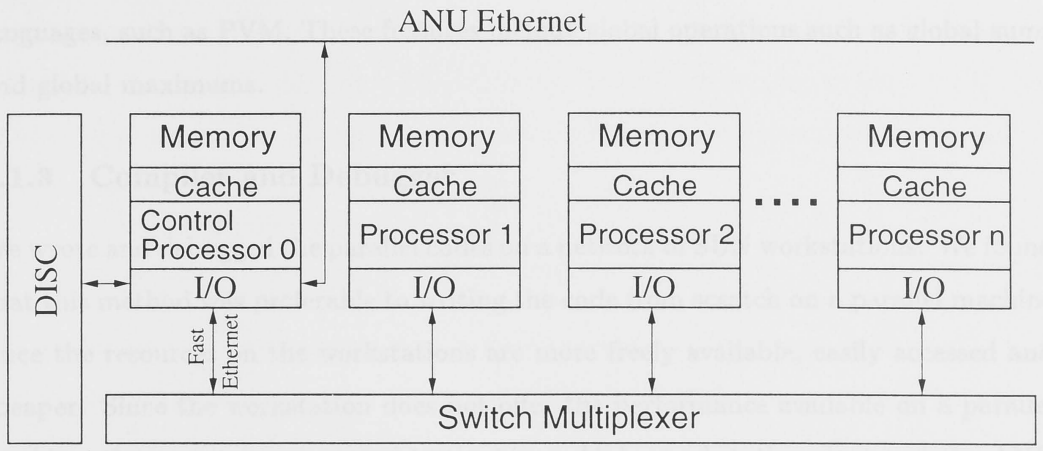

Figure 4-1: Structure of the Linux-Alpha workstation cluster

256MB of memory and 5.3 GBs of IDE disk connected by a HP fast Ethernet switch. An extra node provides compile and file serving facilities using SCSI disks for home directories. Each node has a local scratch partition/scratch on the IDE disk useful for out of core processing. The Alpha was chosen for its price/performance in floating computations - although only marginally more expensive than a 400MHz Pentium II, they perform 1.5 to 2 times faster for floating point intensive codes. Although the HP fast Ethernet is faster than the normal Ethernet, compared with other supercomputers at the ANU Supercomputer Facility, the communication is very expensive on the workstation cluster. Therefore, it is supposed, as will be shown later, that communication is a key factor on determining the partitioning scheme.

4.1.2 Message Passing Interface

The Message Passing Interface (MPI) is a communication library for both parallel computers and workstation networks. MPI has been developed as a proposed standard for message passing and related operations. Its adoption by both users and implementors will provide the parallel programming community with the portability and features needed to develop application programs and parallel libraries that will tap the power of today's high-performance computers.

MPI is a complex system. In its entirety, it comprises 129 functions, many of which have numerous parameters or variants. It is a very user-friendly system. Also, it has many useful features not available in the current version of other parallel programming

languages, such as PVM. These features include global operations such as global sums and global maximums.

4.1.3 Compiler and Debugger

We wrote and debugged the parallel codes on a network of SUN workstations. We found that this method was preferable to writing the code from scratch on a parallel machine since the resources on the workstations are more freely available, easily accessed and cheaper. Since the workstation does not offer the performance available on a parallel machine, the program was ported to the Linux-Alpha workstation cluster at the ANU Supercomputer Facility for performance testing.

On the cluster, we have run codes generated by the COMPAQ F90 Compiler, release 2.91.66.1 compiler, fully compatible with a standard FORTRAN 77 and including advanced FORTRAN 90 features. The debugger we used is GNU gdb 4.17. The resulting codes are compiled with full optimization.

4.2 Performance Evaluation

A sequential algorithm is usually evaluated in terms of its execution time, expressed as a function of the size of its input. The execution time of a parallel algorithm depends not only on the input size but also on the architecture of the parallel computer and the number of processors. Hence, a parallel algorithm cannot be evaluated in isolation from a parallel architecture. A ***parallel system*** is the combination of an algorithm and the parallel architecture on which it is implemented. In this section, we introduce various metrics that are commonly used to measure the performance of parallel systems.

4.2.1 Run time

The serial run time of a program is the time elapsed between the beginning and the end of its execution on a sequential computer. How to measure the ***parallel run time***? The first definition is that the parallel run time is the time that elapses from the moment that a parallel computation starts to the moment that the last processor finishes execution. We denote the serial run time by T_s and the parallel run time by T_p. However, on a multi-user system, the elapsed clock time is dependent on the system

loadings which changes from time to time so that the first definition is inappropriate for measuring the speedup defined in the following. On a time-sharing system , CPU time is more appropriate for analyzing performance and T_p is a measured average among the processors. This suggests the defining

$$T_p = \frac{(Total\ CPU\ time\ for\ allprocesors)}{N} \tag{4.1}$$

is a useful way to assess speedup on a time-sharing multiprocessor system[54]. N is the number of processors used. Since the cluster used in this research is a multi-user system, we used the second definition.

4.2.2 Speedup

When evaluating a parallel system, we are often interested in knowing the performance gain achieved by parallelizing a given application over a sequential implementation. ***Speedup*** is a measure that captures the relative benefit of solving a problem in parallel. It is defined as the ratio of the time taken to solve a problem on a single processor to the time required to solve the same problem on a parallel computer with N identical processors, that is

$$Speedup = \frac{Time\ for\ solution\ on\ 1\ process}{Time\ for\ solution\ on\ N\ processes} \tag{4.2}$$

If we denote speedup by the symbol S, it can be expressed by

$$S = \frac{T_s}{T_p}$$

4.2.3 Efficiency

Only an ideal parallel system containing N processors can deliver a speedup equal to N. In practice, ideal behaviour is not achieved because while executing a parallel algorithm, the processors cannot devote 100 percent of their time to the computations of the algorithm. Part of the time will be required by the interprocessor communication and load imbalance. ***Efficiency*** is a measure of the fraction of time for which a processor is usefully employed. It is defined as the ratio of speedup to the number of processors, i.e.

$$Efficiency = \frac{Speedup}{Number\ of\ processes} \tag{4.3}$$

In an ideal parallel system, speedup is equal to N and efficiency is equal to one. In practice, speedup is less than N and efficiency is between zero and one, depending on the degree of effectiveness with which the processors are utilized. We denote efficiency by the symbol E. Mathematically, it is given by

$$E = \frac{S}{N}$$

4.2.4 Scalability

It is normally the case that the number of processors is an upper bound on the speedup that can be achieved by a parallel system. Speedup is one for a single processor, but if more processors are used, speedup is usually less than the number of processors. Given that increasing the number of processors reduces efficiency and that increasing the size of the computation increases efficiency, it should be possible to keep the efficiency fixed by increasing both the size of the problem and the number of processors simultaneously. This ability to maintain efficiency at a fixed value by simultaneously increasing the number of processors and the size of the problem is exhibited by many parallel systems. We call such systems ***scalable*** parallel systems. The ***scalability*** of a parallel system is a measure of its capacity to increase speedup in proportion to the number of processors. It reflects a parallel system's ability to utilize increasing processing resources effectively.

4.3 Sources of Parallel Overhead

All the sources of performance degradation in a parallel system are a combination of all the causes of inefficiencies of a parallel system, whether due to the algorithm, the architecture, or the algorithm-architecture interaction. The major sources of overhead in a parallel system are interprocessor communication, load imbalance, and extra computation.

4.3.1 Interprocessor Communication

Any nontrivial parallel system requires communication among processors. The time to transfer data between processors is usually the most significant source of parallel processing overhead. If each of the N processors spends the time, t_{comm} for communication, then interprocessor communication contributes $t_{comm} \times N$ to the overhead

function.

4.3.2 Load Imbalance

In many parallel applications, it is impossible to predict the size of the subtasks assigned to various processors. Hence, the problem cannot be subdivided statically among the processors while maintaining an uniform work load. If different processors have different work loads, some processors may be idle during part of the time that others are working on the problem.

Often some or all processors must synchronize at certain points during the parallel program execution. If all processors are not ready for synchronizaton at the same time, then the ones that are ready sooner will be idle until all the rest are ready. Whatever the cause of idling , the total idle time of all the processors contributes to the overhead function.

A special case of overhead due to processor idling is the presence of a sequential component in the parallel algorithm. Part of an algorithm may be unparallelizable, allowing only a single processor to work on it. We express the problem size for such an algorithm as the sum of two components: W_s, the work due to the sequential component, and W_p, the work due to the parallellizable component. While one processor is working on W_s, the remaining $N-1$ are idle. As a result, a serial component of W_s contributes $(N-1)W_s$ to the overhead function of a N-processor parallel system.

4.3.3 Extra Computation

The fastest known sequential algorithm for a problem may be difficult or impossible to parallelize, forcing us to use a parallel algorithm based on a poorer but easily parallelizable sequential algorithm. This will lead to some extra work performed to solve the problem. A parallel algorithm based on the best serial algorithm may still perform more aggregate computation than the serial algorithm. In some cases, the results of certain computation in a serial version can be reused. However, in the parallel version, these results cannot be reused because they are generated by different processors. Therefore, some computations are performed multiple times on different processors. Such extra computations contribute to the overhead function.

4.4 Implementation of Parallel FE Algorithms

To obtain optimal performance from a parallel system it is necessary to tailor the code to the underlying architecture. Such modifications are strongly dependent on the utilized software and hardware. Hence these modifications are usually very specific and can not be carried across from one machine to another. In Chapter 2 and 3, a substepping scheme for integrating the strain-stress relations and a new algorithm for the solution of equations were developed. The issues in the implementation of these algorithms will be considered here. The primary issues for data parallel implementation of the finite element method are:

- Parallel grid generation for transforming the physical domain to discretized computational domain,
- A data structure for representing this computational domain,
- Generation of the elemental stiffness matrices concurrently and
- Concurrent solution of the resulting system of linear equations.

For each of these issues important considerations are

- Storage requirements (uniform storage utilization),
- Communication complexity,
- Parallel arithmetic complexity,
- Programming complexity.

4.4.1 Parallel Grid Generation

The first step involved in the parallel finite element method is to subdivide a finite element domain into a number of subdomains according to the available processors. The objective of such partitioning is to distribute the computational work through the assignment of individual elements or group of elements comprising a portion of the finite element mesh (subdomain) to each processor. It is, in general, well accepted that a domain decomposer should meet at least three basic requirements: (1) it should divide the whole domain into each processor automaticly and involve less communications; (2) it should create subdomains which allow the overall computational load to be as evenly distributed as possible among the processors; (3) it must minimize the amount of interface nodes in order to minimize intersubdomain communication and/or synchronization overhead. To avoid communication at this phase, separate input and

output files are established for each subdomain. These files are read from and written to by local copies of the program executable operating in parallel. After reading corresponding input file, each processor generates the substructure automatically on which it will operate without any need for communication. The advantage of this method is that, each processor only needs to store the data it will use, and this can reduce the storage requirement.

4.4.2 Special numbering scheme

In the example illustrated in the next chapter, a three-dimensional cantilever beam (Figure 2-7) will be used to test the performance of the parallel algorithm developed. The total elements of a large cantilever beam can be numbered in a special way for use on parallel computers. We assumed that a large three-dimensional cantilever beam can be divided evenly by the number of processors to several substructures, so that the common boundary nodes are minimized. The element stiffness matrices in each substructure are then carried out by a single processor.

The method used in [56] is employed in the present study. Numbering the elements in each substructure is achieved in the following way. The element numbers start at the front end, sequentially running over the entire substructure. The far end of a substructure is the front end of the adjoining one. The advantages of this numbering is

- the overlapping problem can be avoided;
- it makes the communications between two joining substructures easy to implement.

4.4.3 Load Balance

Balancing the computation works among the processors is central to achieving high performance. The efficient load balancing is dependent on the assignment of the structure to different processors. Figure 4-2 is an illustration example of a three-dimensional block consisting of $8 \times 8 \times 32$ elements. Three possible partitioning schemes, namely, a vertical strip-wise, a horizontal strip-wise and a box-wise decompositions, are shown respectively in Figure 4-3 - 4-5. It is easy to understand from the Figure 2-7 that, when we increase the nodal forces, the plastic area will begin at the far left end first,

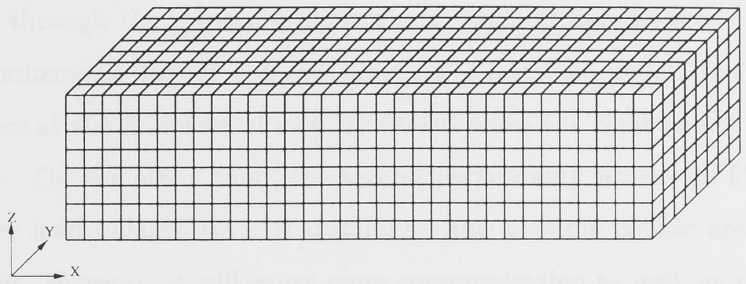

Figure 4-2: A three dimensional cantilever beam ($8 \times 8 \times 32$)

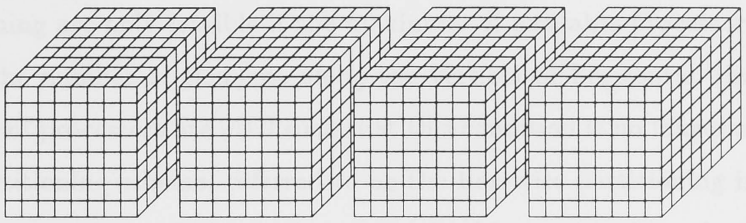

Figure 4-3: A vertical strip-wise partitioning on 4 processors

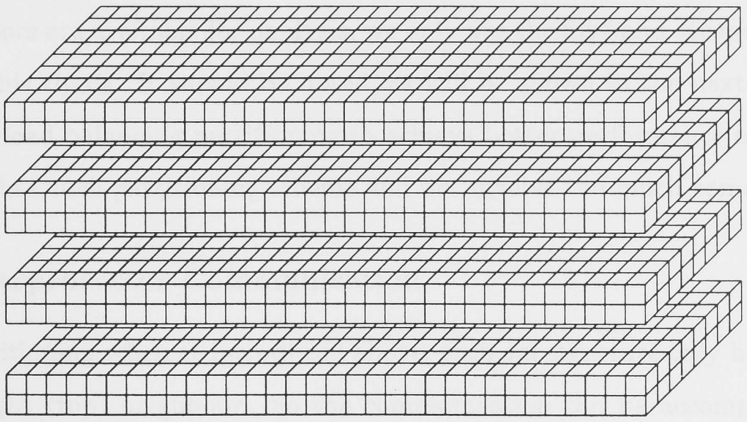

Figure 4-4: A horizontal strip-wise partitioning on 4 processors

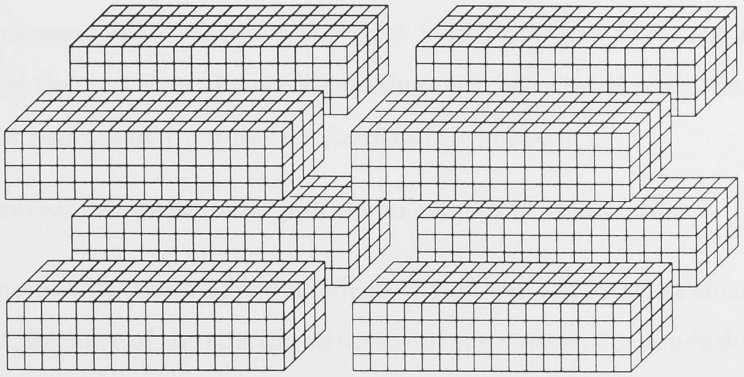

Figure 4-5: A box-wise partitioning on eight processors

and then go through the structure along the x axis. It can be seen that, the first partitioning scheme in Figure 4-3 can reduce the interface nodes, but it can cause load imbalance at the first several load increment when the right end of the structure is still elastic. On the other hand, the second partitioning scheme in Figure 4-4 can lead to better load balance since it distributes the possible plastic area evenly into each processor. However, it will cause more communication as well, as more interface nodes will be involved in the analysis. The vertical strip-wise and a horizontal strip-wise partitioning are only feasible if the maximum of available processors is less than or equal to the number of elements along the vertical or horizontal dimension. If a large number of processors are used such that this condition is no longer satisfied, then the third partitioning scheme, referred to as the box-wise partitioning in Figure 4-5, becomes necessary. It can be seen that not only this partitioning scheme causes the load imbalance, but also it makes the programming relatively complicated. Since only eight processors are available for analysis, we only use the first and second partitioning schemes in this thesis. It should be noted, as will be shown in the next section, that using better load balancing partitioning to achieve better performance must be based on the fact that such partitioning scheme will not involve too much communication.

4.4.4 Interprocessor Communication

For two partitioning schemes introduced above, each substructure only has a left (bottom) and right (top) neighbour. So the communication can be accomplished in two steps as follows:

1. Each processor sends data for the right (top) interface to its right (top) neighbour and then receives the corresponding data associated with the left (bottom) interface being sent from its left (bottom) neighbour.

2. Each processor repeats same process on the opposite direction.

To implement this communication effectively on the workstation cluster, we should be aware that, because of the specific structure of the cluster, the sends do not complete until the matching receives are issued on the destination process. We illustrate this in Figure 4-6. It can be seen that, since at the beginning, only one processor ("top" processor) does not send information to other processors, it can receive from the processor

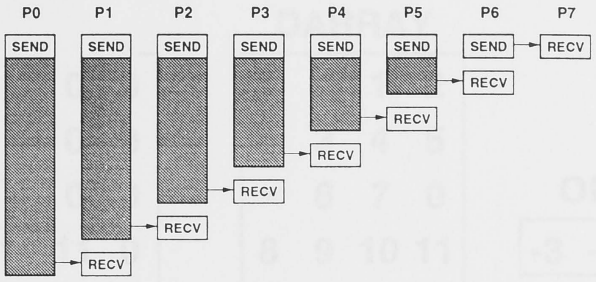

Figure 4-6: Sequentialization caused by sends blocking until the matching receive is posted. The shaded area indicates the time a process is idle.

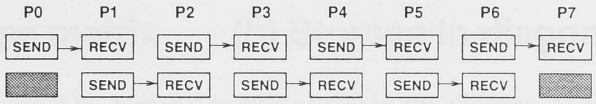

Figure 4-7: Optimizaton of communication by avoiding matching delay.

below it, thus allowing that processor to receive from below it. This illustrates that the program is not executing in parallel. In this thesis, we implement the communication in such a way that the sends and receivers are ordered so that if one process is sending to another, the destination will do a receive that matches that send before doing a send of its own. Figure 4-7 shows the communication pattern using this approach. It can be seen that the even processors send first, and the odd processors receive first. After that, the odd processors send and the even processors receive. This communication process will reduce the idle time caused by the send-receive matching.

4.4.5 Storage Scheme

It is customary to store an $n \times n$ dense matrix in an $n \times n$ array. However, if the matrix is sparse, storage is wasted because a majority of the elements of the matrix are zero and need not be stored explicitly. For sparse matrices, it is a common practice to store only the nonzero entries and to keep track of their location in the matrix. A variety of storage schemes are used to store and manipulate sparse matrixes. These specialized schemes not only save storage but also yield computational savings. Since the locations of the nonzero elements in the matrix are known explicitly, unnecessary multiplications and additions with zero can be avoided. There is no single best data structure for storing sparse matrices. Different data structures are suitable for different operations. Also, some data structures are more suitable for a parallel implementation

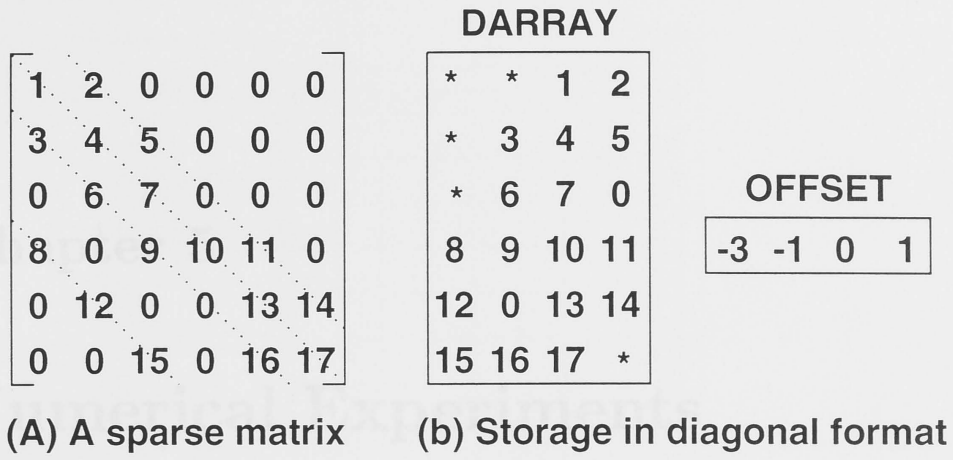

Figure 4-8: A sparse matrix stored in the diagonal format

than others. As the diagonal preconditioner keeps the sparsity of the stiffness matrix, in the following subsections we employ a diagonal storage scheme.

The diagonal storage format is suited to sparse matrices whose nonzero entries are arranged in a few diagonals. Consider an $n \times n$ matrix consisting of d diagonals with nonzero elements (all other entries are zero). These nonzero diagonals are stored in an $n \times d$ array DARRAY. A $d \times 1$ array OFFSET stores the offset of each diagonal with respect to the main diagonal. The order in which the diagonals are stored is not important. Figure 4-8 shows a sparse matrix stored in this fashion. Since all diagonals other than the main diagonal have fewer than n elements, there will be unused locations in the array DARRAY. Any zeros within the d diagonals are stored explicitly.

It can be seen that, if the sparsity is very high, i.e. d is much much less than n, the considerable time wasted on the zero fill-ins can be saved.

Chapter 5

Numerical Experiments

In order to examine the performance of the algorithms developed in this research work, the substepping scheme and the parallel preconditioned conjugate gradient procedure have been applied to the non-linear elasto-plastic finite element analysis of a typical three dimensional cantilever beam. Two structures, shown in Figure 5-1 and 5-2 are analyzed to demonstrate the parallel implementation of the 3-D finite element analysis on the workstation cluster.

5.1 Material Parameters

An elastic-perfect plastic model and an associated flow rule is assumed. The Von Mises yield surface is employed. We use the 8-noded 3D solid element for all analyses. A three-point Gaussian integration rule is considered. The material properties used in the examples are

$$\begin{aligned} E &= 21 \times 10^6 \ psi \\ \nu &= 0.3 \\ {\sigma_Y}^0 &= 24 \times 10^3 \ psi \end{aligned}$$

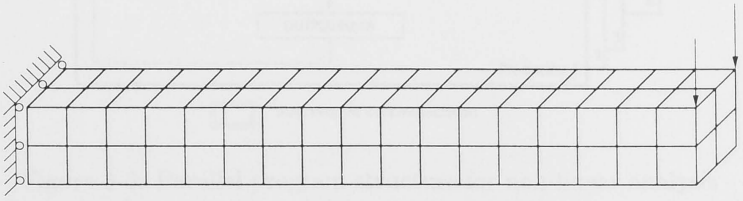

Figure 5-1: 3-D shallow cantilever beam

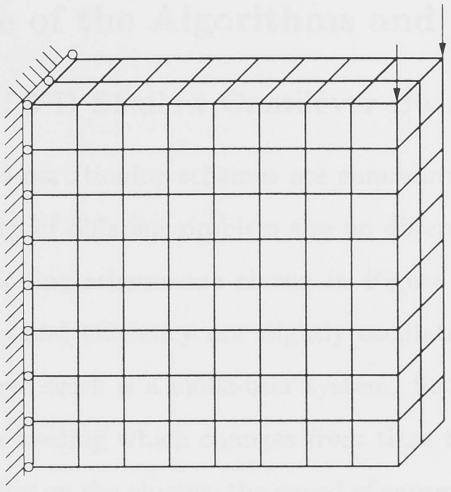

Figure 5-2: 3-D deep cantilever beam

We use a commonly used method for parallel finite element analysis, that is, each processor performs identical instructions, but on different sets of data. The structure of the parallel finite element program is depicted in Figure. (5-3). The parallel implementation of the code has been done via MPI on a Linux-Alpha workstation cluster. The speedup and efficiency are tested for different partitioning schemes introduced in the Chapter 4.

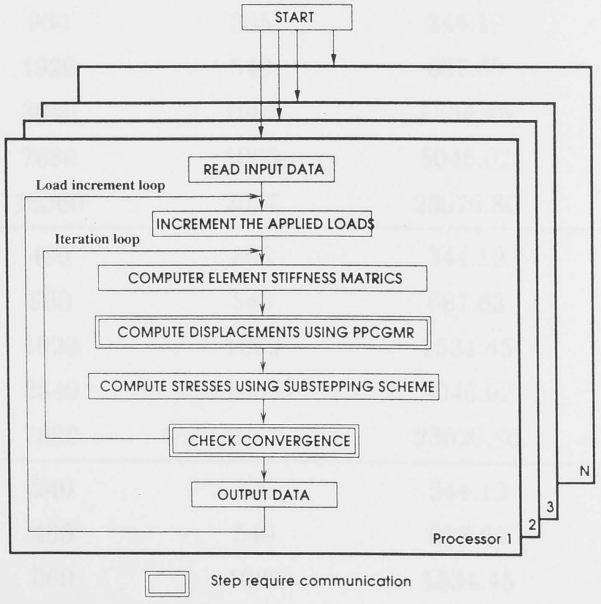

Figure 5-3: Parallel program structure for non-linear analysis

5.2 Performance of the Algorithms and Discussion

5.2.1 Application of 3-D Shallow Cantilever Beam

The characteristics of two partitioning schemes are summarized in Table 5.2 and 5.1. The speedup and efficiency of different problem size on different number of processors by using different partitioning scheme are shown in Figure 5-6 - 5-5. It should be noted that, both speedup and efficiency are slightly oscillatory. This is because the cluster used in this research work is a multi-user system. So the elapsed clock time is dependent on the system loading which changes from time to time. When there are several parallel jobs running on the cluster, the speed of communication will undoubtly slow down.

It is well-known that the speedup and the efficiency will generally increase as the problem size increases. The efficiency of the parallel algorithm generally decreases as

Table 5.1: Substructure of horizontal partitioning scheme for 3-D shallow beam

Number of processors	Elements in each substructure	Interface nodes in each substructure	Elapsed time on 1 processor (sec)	Elapsed time on n processors (sec)
2	960	305	344.19	185.05
	1920	549	687.63	373.71
	3840	1089	1534.45	862.05
	7680	1089	5045.02	2742.36
	15360	2057	23020.80	12648.78
4	480	305	344.19	98.22
	960	549	687.63	204.63
	1920	1089	1534.45	462.18
	3840	1089	5045.02	1404.16
	7680	2057	23020.80	9614.00
8	240	305	344.19	60.19
	480	549	687.63	118.36
	960	1089	1534.45	289.50
	1920	1089	5045.02	801.89
	3840	2057	23020.80	5728.65

Table 5.2: Substructure of vertical partitioning scheme for 3-D shallow beam

Number of processors	Elements in each substructure	Interface nodes in each substructure	Elapsed time on 1 processor (sec)	Elapsed time on n processors (sec)
2	480	25	284.96	152.56
	960	45	1200.71	610.855
	1920	81	2496.63	1280.313
	3840	81	8631.42	4337.396
	7680	153	31875.05	16038.488
	15360	289	79563.56	39802.423
3	320	25	284.96	104.54
	640	45	1200.71	416.608
	1280	81	2496.63	854.640
	2560	81	8631.42	2997.938
	5120	153	31875.05	11343.522
	10240	289	79563.56	28886.052
4	240	25	284.96	83.354
	480	45	1200.71	322.972
	960	81	2496.63	663.873
	1920	81	8631.42	2267.668
	3840	153	31875.05	8726.963
	7680	289	79563.56	20426.663
5	192	25	284.96	67.120
	384	45	1200.71	261.260
	768	81	2496.63	537.594
	1356	81	8631.42	1886.259
	3072	153	31875.05	6838.321
	6144	289	79563.56	17543.590
6	160	25	284.96	58.880
	320	45	1200.71	223.24
	640	81	2496.63	429.747
	1280	81	8631.42	1575.785
	2560	153	31875.05	5692.032
	5120	289	79563.56	14463.87

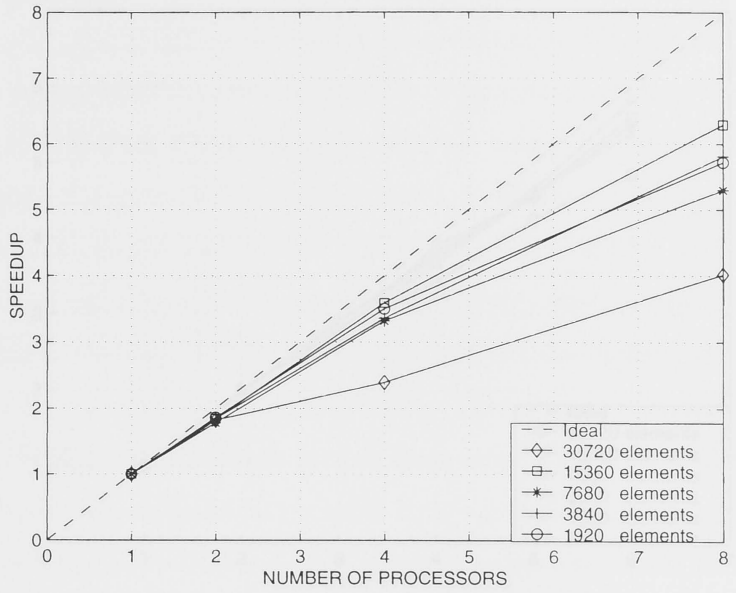

Figure 5-4: Speedup of analyses of 3-D shallow cantilever beam using horizontal strip-wise partitioning

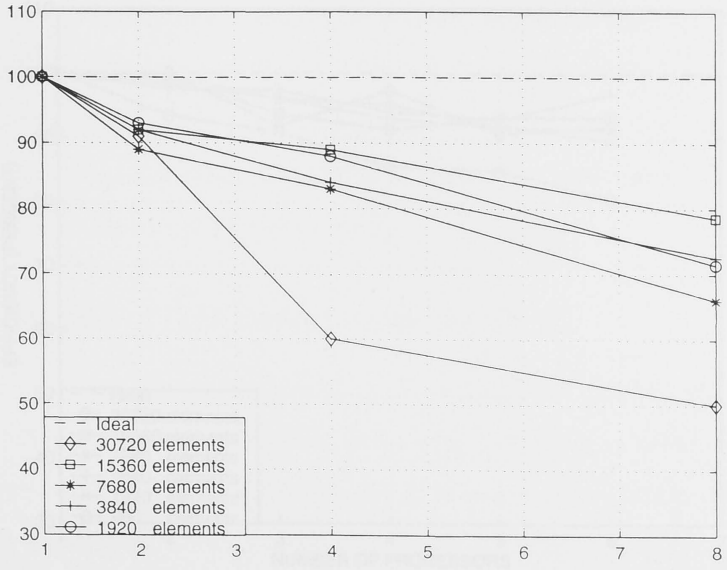

Figure 5-5: Efficiency of analyses of 3-D shallow cantilever beam using horizontal strip-wise partitioning

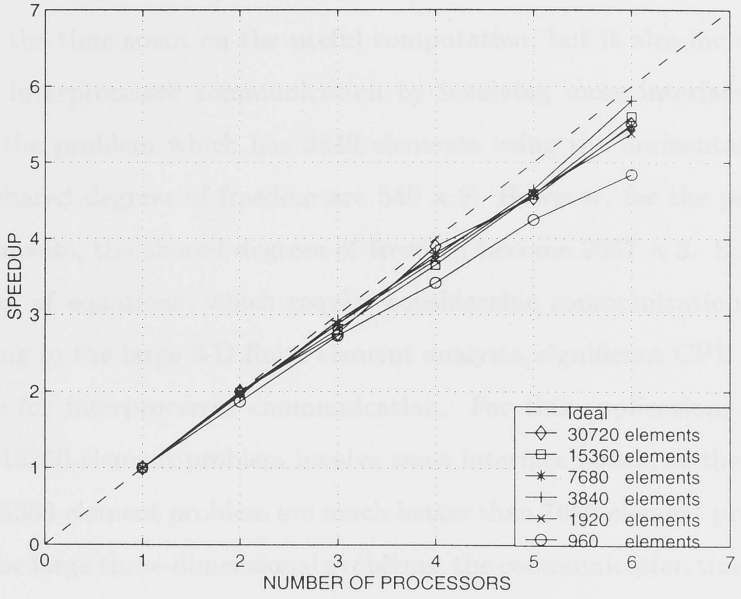

Figure 5-6: Speedup of analyses of 3-D shallow cantilever beam using vertical strip-wise partitioning

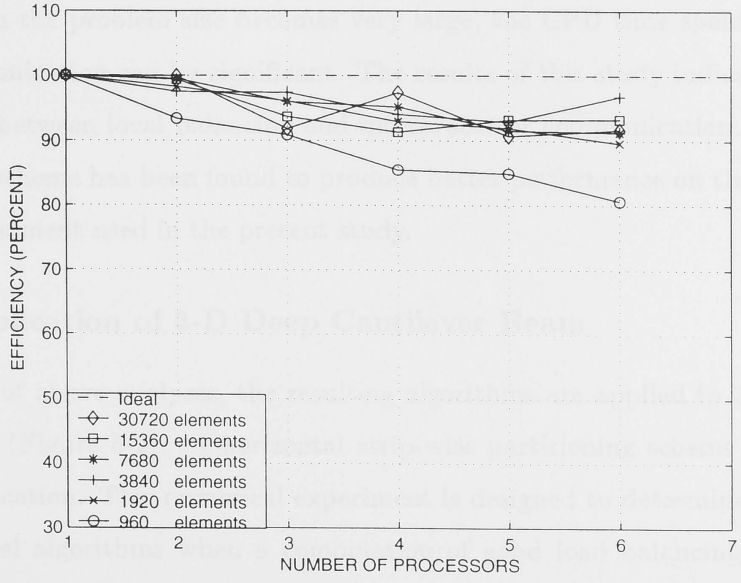

Figure 5-7: Efficiency of analyses of 3-D shallow cantilever beam using vertical strip-wise partitioning

the number of processors increases. This decrease in efficiency is mainly due to the overhead required in the interprocessor communication. Increasing the problem size does increase the time spent on the useful computation, but it also increase the time spent on the interprocessor communication by involving more interface nodes. For example, for the problem which has 3840 elements using the horizontal partitioning scheme, the shared degrees of freedom are 549×3. However, for the problem which has 30720 elements, the shared degrees of freedom become 2057×3. Since the stage of the solution of equations, which require considerable communication, is the most time-consuming in the large 3-D finite element analysis, significant CPU time is spent on this phase for interprocessor communication. For this application, 7680-element problem and 15360-element problem involve same interface nodes, so the speedup and efficiency of 15360-element problem are much better than 7680-element problem. It can be seen that for large three-dimensional problems, the communication time is one of the dominant factors on the workstation cluster. This is one of the reasons for reduction in performance with the increases in problem sizes.

It can also be seen that the horizontal partitioning scheme leads to a better load balancing due to the different processor involvement in the plastic calculation. However the vertical partitioning scheme leads to a reduction in the interprocessor communication. When the problem size becomes very large, the CPU time spent on interprocessor communication can be significant. The results of this study indicate that there is a tradeoff between local balancing and interprocessor communication. The vertical partitioning scheme has been found to produce better performance on the workstation cluster environment used in the present study.

5.2.2 Application of 3-D Deep Cantilever Beam

On the base of above analyses, the resulting algorithms are applied to 3-D deep cantilever beam (Figure 5-2). A horizontal strip-wise partitioning scheme is considered for this application. This numerical experiment is designed to determine performance of the parallel algorithms when a combination of good load balancing and reduced interprocessor communication is employed.

The characteristics of horizontal partitioning scheme for 3-D deep beam are summarized in Table 5.3. The speedup and efficiency of different problem size on different

number of processors are shown in Figure 5-8 and 5-9. From the results we can see that the performance of horizontal partitioning scheme in this application is better than for the shallow beam. In fact, even for small problem size (i.e. 960 elements), almost perfect results are obtained. This perfect results can also be attributed to the employment of substepping scheme which can integrate the strain-stress relations accurately and thus make the solution more efficient. It again clarifies that, to get an optimal performance on the parallel systems like the workstation cluster on which the communication is very expensive, adoption of the partitioning scheme must be based on the fact that such partitioning scheme involves least amount of interprocessor communication.

Table 5.3: Substructure of horizontal partitioning scheme for 3-D deep beam

Number of processors	Elements in each substructure	Interface nodes in each substructure	Elapsed time on 1 processor (sec)	Elapsed time on n processors (sec)
2	480	25	87.70	44.52
	960	45	360.62	184.31
	1920	81	746.47	375.50
	3840	153	1790.85	898.20
	7680	289	5892.09	3015.00
	15360	561	28936.26	14817.03
3	320	25	87.70	29.53
	640	45	360.62	126.33
	1280	81	746.47	276.80
	2560	153	1790.85	624.97
	5120	289	5892.09	1967.03
	10240	561	28936.26	10059.02
4	240	25	87.70	22.23
	480	45	360.62	92.64
	960	81	746.47	191.25
	1920	153	1790.85	460.38
	3840	289	5892.09	1511.71
	7680	561	28936.26	7342.32
5	192	25	87.70	17.96
	384	45	360.62	82.34
	768	81	746.47	156.99
	1356	153	1790.85	400.35
	3072	289	5892.09	1189.82
	6144	561	28936.26	6012.85
6	160	25	87.70	15.20
	320	45	360.62	66.09
	640	81	746.47	128.92
	1280	153	1790.85	321.35
	2560	289	5892.09	1023.00
	5120	561	28936.26	4823.71

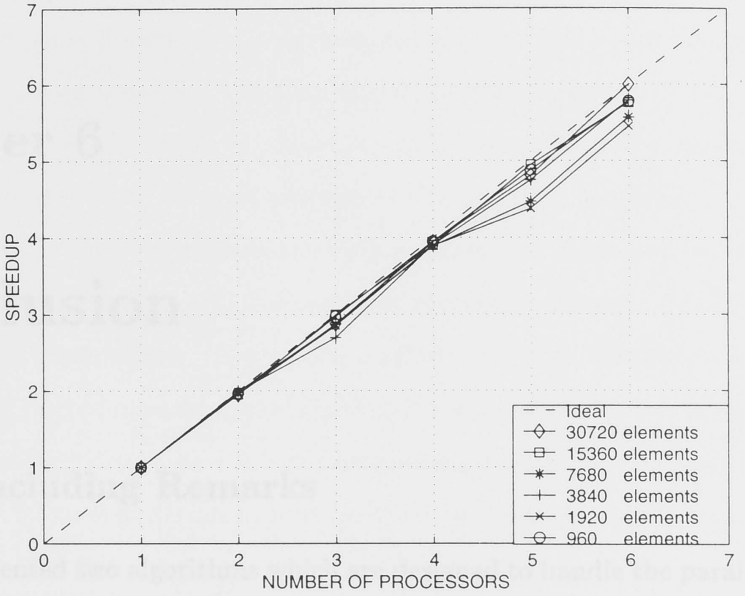

Figure 5-8: Speedup of analyses of 3-D deep cantilever beam

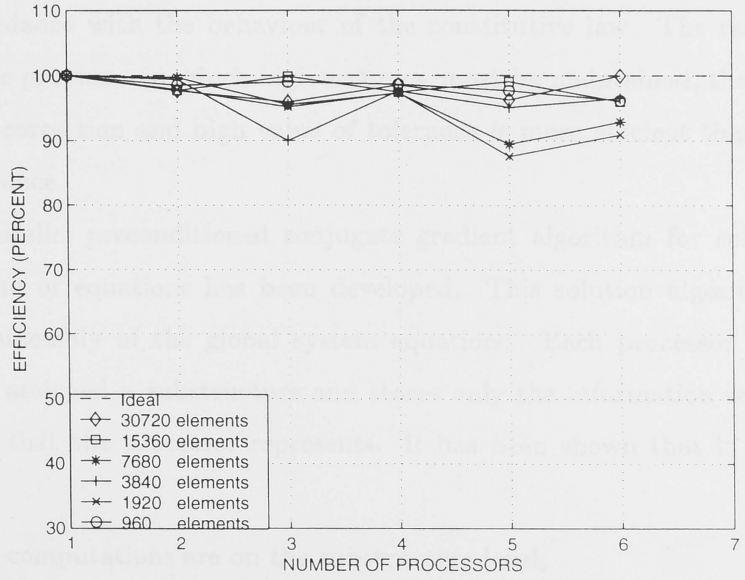

Figure 5-9: Efficiency of analyses of 3-D deep cantilever beam

Chapter 6

Conclusion

6.1 Concluding Remarks

We have presented two algorithms which are designed to handle the parallel implementation of elasto-plastic problems using the finite element method.

An advanced substepping scheme has been developed for the elasto-plastic stress analysis. This substepping scheme based on the Gill's fourth-order Runge-Kutta method controls the error in the integration process to the vicinity of a specified tolerance. This mechanism for controlling the integration process permits the size of each substep to vary in accordance with the behaviour of the constitutive law. The results indicate that, for large problems in which high accuracy must be maintained, the combination of the stress correction and high value of tolerance is more efficient than using small value of tolerance.

A new parallel preconditioned conjugate gradient algorithm for solving the linearized system of equations has been developed. This solution algorithm does not require the assembly of the global system equations. Each processor in the parallel system is assigned a substructure and stores only the information relevant to the substructure that the processor represents. It has been shown that it has following advantages:

- All the computations are on the substructure level;
- Improves the rate of convergence of CG algorithm;
- Saves memory.

The combination of these two algorithms have been applied to a typical three dimensional elasto-plastic stress analysis. Several techniques have been used with an aim to get optimal performance on the workstation cluster. The results indicate that, if a suitable partitioning scheme is used, an almost perfect performance can be produced. In summary, the combination of these two algorithm provides a powerful practical strategy for parallel finite element analysis of elasto-plastic problems.

The results obtained in this research work indicate that some load balancing schemes can degrade the performance if the partitioning scheme involves large amount of interprocessor communication. It reminds us that, although idle time caused by poor load balancing degrades performance quickly, if the parallel analysis involves too much communication by using some 'perfect' partitioning schemes, the advantage of load balancing caused by such partitioning may be offset by the interprocessor communication.

6.2 Future Work

Research and development of industrial manufacturing processes and products typically require lengthy and extensive prototype testing and experimentation in arriving at a competitive product. High performance computing systems directly affect the computational time and the level of fine details that can be incorporated into the models, and hence the accuracy and benefits that can be obtained. Therefore, there is a need to develop an effective high performance computational strategy for performing an entire finite element solution procedure with relevance to manufacturing applications. In order to fully understand the physics of the manufacturing processes and its effect on product performance and cost, the following areas need to be researched:

- Automated parallel finite element mesh generation schemes for three dimensional geometries;
- Parallel Finite Element formulation that account for material and geometrical nonlinearities;
- Contact and Friction models and their efficient parallel implementation in the finite element procedure;
- Parallel adaptive mesh generation during the simulation of the manufacturing process to account for elemental distortion and for improved accuracy in the solution procedure;

- The integration of the above feature to provide a unified high performance computational strategy for modelling manufacturing processes.

The above features of future research work will lead to a better understanding of the physics involved in process modelling and when combined with a robust optimisation algorithm will provide the designer a powerful tool in designing high quality products at a faster rate and reduced cost.

Bibliography

[1] G. C. Nayak and O. C. Zienkiewicz. Elastoplastic Stress Analysis: A Generation for Various Constitutive Laws Including Strain Softening. *Int, J. Numer. Methods Eng.*, 5:113-135, 1972.

[2] D. R. J. Owen and E. Hinton. Finite Element in Plasticity: Theory and Practice. Pineridge Press, Swansea, 1980.

[3] D. M. Potts and A. Gens. A Critical Assessment of Methods of Correcting for Drift from the Yield Surface in Elasto-Plastic Finite Element Analysis. *International Journal For Numerical And Analytical Methods In Geomechanics*, 9:149–159, 1985.

[4] J. W. Wissmann and C. Hauck. Efficient Elastic-Plastic Finite Element Analysis with Higher Order Stress-Point Algorithms. *Comput. Struct.*, 17:89–95, 1983.

[5] M. U. Polat and M. A. Dokainish. An Automatic Subincrementation Scheme for Accurate Integration of Elasto-Plastic Constitutive Relations. *Computers & Structures* , 31:339-347, 1989.

[6] A. L. Eterovic and K-J Bathe. A Hyperelastic-Based Large Strain Elasto-Plastic Constitutive Formulation with Combined Isotropic-Kinematic Hardening Using the Logarithmic Stress and Strain Measures. *International Journal For Numerical Methods In Engineering*, 30:1099-1114, 1990.

[7] S. Pezeshk and C. V. Camp. An Explicit Time Integration Technique for Dynamic Analyses. *International Journal For Numerical Methods In Engineering*, 38:2265-2281, 1995.

[8] S. W. Sloan. Substepping Schemes for the Numerical Integration of Elastoplastic Stress-Strain Relations. *International Journal For Numerical Methods In Engineering*, 24:893–911, 1987.

[9] S. W. Sloan and J. R. Booker. Integration of Tresca and Mohr-Coulomb Constitutive Relations in Plane Strain Elastoplasticity. *International Journal For Numerical Methods In Engineering*, 33:163-196, 1992.

[10] A. Gens and D. M. Potts. Critical State Models in Computational Geomechanics. *Eng. Comput.*, 5:178–196, 1988.

[11] J. W. Joo and B. M. Kwak. Analysis and Applications of Elasto-Plastic Contact Problems Considering Large Deformation. *Computers & Structures*, 24:953–961, 1986.

[12] J. B. Martin and W. W. Bird. Integration Along the Path of Loading in Elastic-Plastic Problems. *Eng. Comput.*, 5:217–223, 1988

[13] H. Jin, K. Mattiasson, K. Runesson and A. Samuelsson . On The Use of the Boundary Element Method for Elastoplastic, Large Deformation Problems. *International Journal For Numerical Methods In Engineering*, 25:165–176, 1988.

[14] M. A. Dokainish and K. Subbaraj. A Survey of Direct Time-Integration Methods in Computational Structural Dynamics-I. Explicit Methods. *Computers & Structures* 32:1371-1386, 1989.

[15] X. Deng and A. J. Rosakis. Negative Plastic Flow and Its Prevention in Elasto-Plastic Finite Element Computation. *Finite Elements in Analysis and Design*, 7:181-191, 1990.

[16] Matti Ristinmaa and Johan Tryding. Exact Integration of Constitutive Equations in Elasto-Plasticity. *International Journal For Numerical Methods In Engineering*, 36:2525-2544, 1993.

[17] X. Peng and J. Fan. A Numerical Approach for Nonclassical Plasticity. *Computers & Structures*, 47:313-320, 1993.

[18] A. Ranjbaran and M. E. Phipps. Dena: A Finite Element Program for the Non-Linear Stress Analysis of Two-Dimensional, Metallic and Reinforced Concrete, Structures. *Computers & Structures*, 51:191-211, 1994.

[19] M. Ristinmaa. Consistent Stiffness Matrix in FE Calculations of Elasto-Plastic Bodies. *Computers & Structures*, 53:93-103, 1994.

[20] F. Auricchio and R. L. Taylor. A Generalized Elastoplastic Plate Theory and Its algorithmic Implementation. *International Journal For Numerical Methods In Engineering*, 37:2583-2608, 1994.

[21] J. Noorzaei, M. N. Viladkar and P. N. Godbole. Elasto-Plastic Analysis for Soil-Structure Interaction in Framed Structures. *Computers & Structures*, 55:797-807, 1995.

[22] Stefan M. Holzer and Zohar Yosibash. The P-Version of the Finite Element Method in Incremental Elasto-Plastic Analysis. *International Journal For Numerical Methods In Engineering*, 39:1859-1878, 1996.

[23] vinod K. Arya. Efficient and Accurate Explicit Integration Algorithms with Application to Viscoplastic Models. *International Journal for Numerical Methods in Engineering*, 39:261-279, 1996.

[24] H. Matthies and G. Strang. The Solution of Nonlinear Finite Element Equations. *Int. J. Numer. Methods Eng.*, 14:1613-1626, 1979.

[25] Chang-Koon Choi and Heung-Jin Chung. Error Estimates and Adaptive Time Stepping for Various Direct Time Integration Methods. *Computers & Structures*, 60:923-944, 1996.

[26] Jacob Fish and Kamlun Shek. Computational Aspects of Incrementally Objective Algorithms for Large Deformation Plasticity. *International Journal For Numerical Methods In Engineering*, 44:839-851, 1999.

[27] G. De Roeck, M. Van Laethem and Sheu Chyi-Horng. Multi-Level Substructuring in the Elasto-Plastic Domain. *Computers & Structures* , 31:757-765, 1989.

[28] O. C. Zienkiewicz and R. L. Taylor. The Finite Element Method. McGraw-Hill Book Company, Fourth Edition, Volume 1, 1991.

[29] O. C. Zienkiewicz and R. L. Taylor. The Finite Element Method. McGraw-Hill Book Company, Fourth Edition, Volume 2, 1991.

[30] M. L. James, G. M. Smith and J. C. Wolford. Applied Numerical Methods for Digital Computation. Harper & Row, Publisher, New York, 1985.

[31] E. Hinton and D. R. J. Owen. *Finite Element Programming.* Academic Press Inc. London, 1977.

[32] H. Kardestuncer. Finite Element Handbook. McGraw-Hill, Inc. 1987.

[33] E. Wilson and C. Farhat. Linear and Nonlinear Finite Element Analysis on Multiprocessor Computer Systems. *Commun. Appl. Numer. Meth.*, 4:425-434, 1988.

[34] C. T. Sun and K. M. Mao. Elastic-Plastic Crack Analysis Using A Global-Local Approach on a Parallel Computer. *Computers & Structures*, 30:395-401, 1988.

[35] C. Farhat and L. Crivell. A General Approach to Nonlinear FE Computations on Shared-Memory Multiprocessors. *The Methods in Applied Mechanics and Engineering*, 72:153-171, 1989.

[36] K. N. Shivakumar, C. A. Bigelow and J. C. Newman,Jr. Parallel Computation in a Three-Dimensional Elastic-Plastic Finite-Element Analysis. *Computers & Structures*, 43:237-245, 1992.

[37] S. kacou and I. D. Parsons. A Parallel Multigrid Method for History-Dependent Elastoplasticity Computations. *Computer Methods in Applied Mechanics and Engineering*, 108:1-21, 1993.

[38] Ning Hu. A Parallel Algorithm for Analyzing Elasto-Plastic Problems. *Computers & Structures*, 52:1127-1133, 1994.

[39] S. Kacou and I. D. Parsons. A Parallel Multigrid Method for History-Dependent Elastoplasticity Computations. *Engineering Computation*, 11:347-355, 1994.

[40] A. Feriani, A. Franchi and F. Genna. An Incremental Elastic-Plastic Finite Element Solver in a Workstation Cluster Environment Part 1. Formulations and Parallel Processing. *Comput. Methods Appl.Mech. Engrg.*, 130:289-298, 1996.

[41] A. Feriani, A. Franchi and F. Genna. An Incremental Elastic-Plastic Finite Element Solver in a Workstation Cluster Environment Part 2. Performance of a First Implementation. *Comput. Methods Appl. Mech. Engrg.*, 130:299-318, 1996.

[42] R. J. Melosh and S. Utku. Direct Finite Element Equation Solving Algorithms. *Comput. Struct.*, 8:621-632, 1977.

[43] S. Doi and S. Koyama. A Parallel Computation Technique for Finite Element Methods. *Systems Comp. Controls*, 13:76-84, 1982.

[44] A. K. Noor, H. A. Kamel and R. E. Fulton. Substructuring Technique-Status and Projections. *Comput. Struct.*, 29:99–105, 1985.

[45] C. Farhat and E.Wilson. A New Finite Element Concurrent Computer Program Architecture. *International Journal for Numerical Methods In Engineering*, 24:1771-1792, 1987.

[46] J. G. Malone. Automated Mesh Decomposition and Concurrent Finite Element Analysis for Hypercube Multiprocessor Computers. *Comp. Meth. Appl. Mech. Engng.*, 70:27-58, 1988.

[47] D. Goehlich, L. Komzsik and R. E. Fulton. A New Finite Element Concurrent Computer Program Architecture. *International Journal for Numerical Methods In Engineering*, 24:1771-1792, 1987.

[48] T. J. R. Hughes, I. Levit and J. Winget. An Element-by-Element Solution Algorithm for Problems of Structural and Solid Mechanics *Comput. Meths. Appl. Mech. Engrg*, 36:245-254, 1986.

[49] Kincho H. Law. A Parallel Finite Element Solution Method *Computers & Structures*, 23:845-858, 1986.

[50] R. B. King and K. Sonnad. Implementation of An Element-by-Element Solution Algorithm for the Finite Element Method on A Coarse-grained Parallel Computer *Comput. Meths. Appl. Mech. Engrg*, 65:47-59, 1987.

[51] W. T. Carter, Jr, T.-L. Sham and Kincho H. Law. A Parallel Finite Element Method and Its Prototype Implementation on A Hypercube *Computers & Structures*, 31:921-934, 1989.

[52] S. L. Johnsson and K. K. Mathur. Experience with the Conjugate Gradient Method for Stress Analysis on a Data Parallel Supercomputer *International Journal for Numerical Methods in Engineering*, 27:523-546, 1989.

[53] V. Kumar, A. Grama, A. Gupta and G. Karypis. Introduction to Parallel Computing: Design and Analysis of Algorithms. The Benjamin/Cumings Publishing Company, Inc. 1994.

[54] J-C Luo and M. B. Frienman. A Parallel Computational Model for the Finite Element Method on a Memory-Sharing Multiprocessor Computer. *Computer Methods in Applied Mechanics and Engineering*, 84:193-209, 1990.

[55] David L. Herendeen. Parallel Processing and FEM: Fulfilling the Promise. *Finite Elements in Analysis and Design*, 4:193-202, 1988.

[56] L. S. Chien and C. T. Sun. Parallel Processing Techniques for Finite Element Analysis of Nonlinear Large Truss Structures. *Computers & Structures*, 31:1023-1029, 1989.

[57] S. L. Johnsson and K. K. Mathur. Data Structures and Algorithms for the Finite Element Method on A Data Parallel Supercomputer. *International Journal for Numerical Methods in Engineering*, 29:881-908, 1990.

[58] G. Yagawa, N. Soneda and S. Yoshimura. A Large Scale Finite Element Analysis Using Domain Decomposition Method on a Parallel Computer. *Computers & Structures*, 38:615-625, 1991.

[59] K. K. Yalamanchili, S. C. Anand and D. D. Warner. Three-Dimensional Finite Element Analysis on a Hypercube Computer. *Computers & Structures*, 42:11-20, 1992.

[60] Charbel Farhat and Michel Lesoinne. Automatic Partitioning of Unstructured Meshes for the Parallel Solution of Problems in Computational Mechanics. *International Journal for Numerical Methods in Engineering*, 36:745-764, 1993.

[61] M. W. S. Jaques, C. T. F. Ross and P. Strickland. Exploiting Inherent Parallelism in Non-Linear Finite Element Analysis. *Computers & Structures*, 58:801-807, 1996.

[62] David R. Oakley, Norman F. Knight and Jr.. Non-Linear Structural Response Using Adaptive Dynamic Relaxation on A Massively Parallel-Processing System. *International Journal for Numerical Methods in Engineering*, 39:235-259, 1996.

[63] T. Coupez and S. Marie. From A Direct Solver to a Parallel Iterative Solver in 3-D Forming Simulation. *The International Journal of Supercomputer Applications and High Performance Computing*, 11:277-285, 1997.

[64] E. J. Plaslacz. Parallel Finite-Element Analysis via Message Passing. *Microcomputers in Civil Engineering*, 12:101-118, 1997.

[65] I. ST. Doltsinis and Swen Nolting. Parallel Algorithms for the Modelling and Simulation of Industrial Metal Forming. *Numerical Methods in Industrial Forming Processes*, 35-43, 1992.

[66] I. Foster. Designing and Building Parallel Programs. Addison-Wesley Publishing Company,1995.

[67] J. Miklosko and V. E. Kotov. Algorithms, Software and Hardware of Parallel Computers. Springer-Verlag and VEDA, Czechoslovakia, 1984.

[68] William Gropp, Ewing Lusk and Anthony Skjellum. *USING MPI Portable Parallel Programming with the Message-Passing Interface.* The MIT Press, Cambridge, Massachusetts, London, England,1994.

[69] T. G. Lewis. Foundation of Parallel Programming. IEEE Computer Society Press, Los Alamitos, California, 1993.

[70] B. P. Sommeijer. *Parallelism in the Numerical Integration of Initial value Problems.* Stichting Mathematisch Centrum, Amsterdam, 1993

[71] J. A. Zonneveld. *Automatic Numerical Integration.* Stichting Mathematisch Centrum, Amsterdam, 1970.

[72] J. M. Levesque and J. W. Williamson. A Guidebook to Fortran on Supercomputers. Academic Press, Inc. 1989.

[73] G. V. Wilson. Practical Parallel Programming. The MIT Press, Cambridge, Massachusetts, London, England, 1995.

[74] R. H. Wagoner and J-L. Chenot. Fundamental of Metal Forming. John Wiley & Sons, Inc. 1996.

Appendix A

Runge-Kutta Methods

A Runge-Kutta Method is one which employs a recurrence formula of the form

$$y_{i+1} = y_i + a_1 k_1 + a_2 k_2 + a_3 k_3 + \cdots + a_n k_n \tag{A.1}$$

to calculate successive values of the dependent variable y of the differential equation

$$\frac{dy}{dx} = y' = f(x, y) \tag{A.2}$$

where

$$\begin{aligned} k_1 &= (h) f(x_i, y_i) \\ k_2 &= (h) f(x_i + p_1 h, y_i + q_{11} k_1 \\ k_3 &= (h) f(x_i + p_2 h, y_i + q_{21} k_1 + q_{22} k_2) \\ &\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ k_n &= (h) f(x_i + p_{n-1} h, y_i + q_{n-1,1} k_1 + q_{n-1,2} k_2 + \cdots + q_{n-1,n-1} k_{n-1}) \end{aligned} \tag{A.3}$$

The above Runge-Kutta equations can be written more compactly as

$$y_{i+1} = y_i + \sum_{j=1}^{n} a_j k_j \tag{A.4}$$

where

$$k_j = h f(x_i + p_{j-1} h, y_i + \sum_{l=1}^{j-1} q_{j-1,l} k_l) \quad (j = 1,\ 2, \cdots,\ n) \tag{A.5}$$

in which, by definition,

$$p_0 = 0 \quad and \quad \sum_{l=1}^{j-1} q_{j-1,l} k_l = 0 \quad j = 1 \tag{A.6}$$

The $a's$, $p's$, and $q's$ must assume values such that Equation A.1 accurately yields successive values of y. These values are determined by making Equation A.1 equivalent to a certain specified number of terms of a Taylor-series expansion of y about x_i.

If we let m be the order of the Runge-Kutta method, with n the number of function evaluations per step (number of k values), there is a particular maximum value of m for each n. For n values up to 7, we have

$$\begin{aligned} 1 \leq n &\leq 4, \; m_{max} = n \\ n &= 5, \; m_{max} = 4 \\ n &= 6, \; m_{max} = 5 \\ n &= 7, \; m_{max} = 6 \end{aligned}$$

Various Runge-Kutta methods are classified as (m, n) methods. A $(5, 6)$ method, for example, would be a fifth-order method requiring six function evaluations per step.